ISSN 0868-6157
Совместное советско-американское предприятие «СОВАМИНКО»

КОМПЬЮТЕР ПРЕСС

ОБОЗРЕНИЕ ЗАРУБЕЖНОЙ ПРЕССЫ

КОМПЬЮТЕРНЫЕ ВИРУСЫ:
предварительные
соображения

5'91

It's CeBIT Time

HANNOVER MESSE
CeBIT '91
13. – 20. MÄRZ 1991

Совместное советско-американское предприятие «СОВАМИНКО»

КОМПЬЮТЕР ПРЕСС

ОБОЗРЕНИЕ ЗАРУБЕЖНОЙ ПРЕССЫ



5'91

Совместное советско-американское предприятие «СОВАМИНКО»

КОМПЬЮТЕР ПРЕСС

ОБОЗРЕНИЕ ЗАРУБЕЖНОЙ ПРЕССЫ

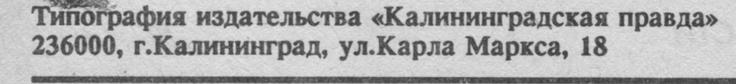
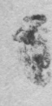
Адрес редакции:
113093, г.Москва, аб.ящик 37
Тел. для справок: 150-17-03
Бюро рекламы: 156-81-33
Факс: 200-22-89
E-mail:
postmaster@Computerpress.msk.su

Ты побывал на Comtek'91?

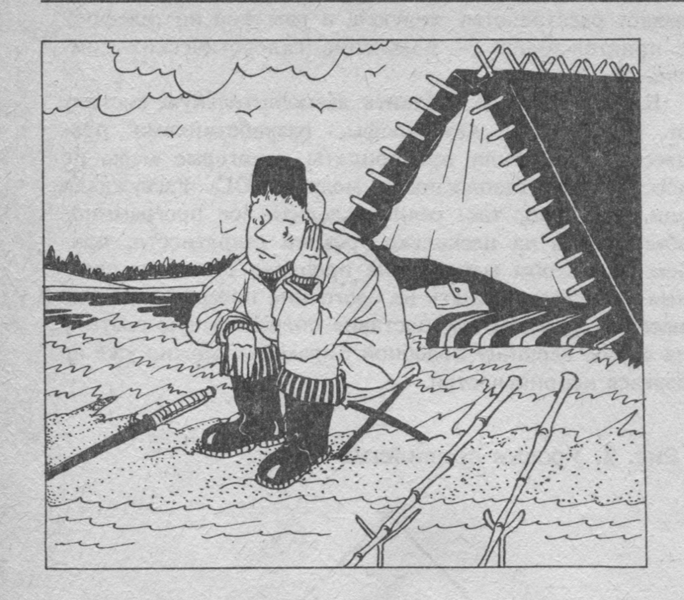
Если нет, то напрасно. Ведь общение было и остается одним из самых мощных стимулов прогресса. Безусловно, лишь в том случае, когда собеседник понимает тебя с полуслова, когда разговор ведут профессионалы. На Comtek разговор велся между фирмами, определяющими лицо компьютерной индустрии. Поэтому результаты выставки обязательно отразятся на дальнейшем развитии этой отрасли.

Если все же тебе не удалось побывать на Comtek'91, не отчаивайся. "КомпьютерПресс" постарается восполнить эту потерю. В этом номере мы начинаем публикацию серии материалов о выставке и интервью с ее участниками.

Сдано в набор 10.04.91. Подписано к печати 28.04.91. Формат 84x108/16. Печать офсетная.
Усл.печ.л.8,4+0,32 (обл.). №028. Тираж 100 000 экз. (1 завод–55 000). Заказ № 2260. Цена 3 р. 15 к.

Типография издательства «Калининградская правда»
236000, г.Калининград, ул.Карла Маркса, 18



Эта статья завершает цикл, посвященный архитектуре шестнадцатиразрядных процессоров фирмы Intel. В своем развитии они прошли все этапы — от робкого появления на международном рынке, до полного расцвета и обидно быстрого ухода со сцены. Им на смену пришли могучие 32-битные кристаллы, уже сегодня занявшие практически все "экологические ниши" в компьютерной индустрии. В следующих номерах "КомпьютерПресс" мы постараемся как можно интереснее рассказать об их интимной, скрытой от постороннего взгляда жизни.

Архитектура микропроцессоров

Совершенно секретно, перед прочтением сжечь!

А. и Б. Стругацкие

Аналогии, аналогии...

Наверное, я не ошибусь, если скажу, что каждый из нас хотя бы раз в жизни слышал из уст "ответственных" работников фразу: "Это вам знать не полагается!" — и немудрено, ведь тайны окружают человека на протяжении всей жизни. Посудите сами — сначала родители, блюдя нравственность своего чада, плетут ему байки про аиста и капусту, затем, уже в школе, на него заводится совершенно секретное личное дело, а потом он попадает во взрослый мир, где таинственные и легендарные тени КГБ, МВД и ВПК грозно реют над несчастным любознательным индивидуумом до конца его дней. Но вот парадокс: соблюдение своих секретов и разгадывание чужих — эта чрезвычайно сложная и очень дорогая игра для серьезных дяденек — приносит, в основном, сугубо отрицательный результат. Мало того, что на обмундирование и прокорм всей этой оравы уходит масса средств, так оказывается и тайны-то отечественные остаются таковыми только для местных жителей, а разгадка чужих требует не нашей технологии и не наших знаний.

Другое дело тайны компьютерные. Многолетний опыт показывает, что на создание секретных зон в ОЗУ потратиться стоит. У некоторых читателей подобный тезис может вызвать удивление, но если вдуматься, действительно начинаешь сомневаться, весь ли объем оперативной памяти машины должен быть доступен для любого пользователя? Давайте попробуем в этом деле разобраться.

Берегите ОС

Грубо говоря, программное обеспечение, расположенное в ОЗУ, можно разделить на две основные части: операционную систему (ОС) и прикладные программы. Первая, как вам известно, служит для управления аппаратными ресурсами компьютера и осуществления взаимодействия между прикладными программами, пользователем и "железом". Она явля-

ется как бы диспетчером для всех процессов, происходящих в вычислительной системе. Если провести параллель между компьютером и человеческим организмом, то ОС в большинстве случаев выполняет функции вегетативной, а прикладные программы — центральной нервной системы.

Теперь представьте себе на минутку, что написанная вами программа случайно внесла какие-то изменения в один из сегментов ОС. Это почти наверняка приведет к нарушению таких жизненно важных для компьютера функций, как управление памятью, вводом-выводом или файловой системой. Продолжая аналогию с человеком, практически невозможно представить себе ситуацию, когда чтение утренней газеты вызывает расстройство желудка, а разговор по телефону с приятельницей — появление склеротических симптомов.

Для того чтобы избавить вычислительную систему от неминуемой катастрофы, разработчиками процессора i286 были предприняты некоторые меры по обеспечению безопасности модулей ОС. Рассуждали они, очевидно, так: если разделить все программное обеспечение на несколько уровней секретности, причем так, чтобы выполнение процедур с низким уровнем не могло повлиять на состояние программ с более высоким уровнем, то, поставив операционную систему на самую вершину подобной пирамиды, можно уже не бояться неприятностей.

Рис. 1. Классификация секретности документов в США

Рис. 2. Уровни привилегий i286

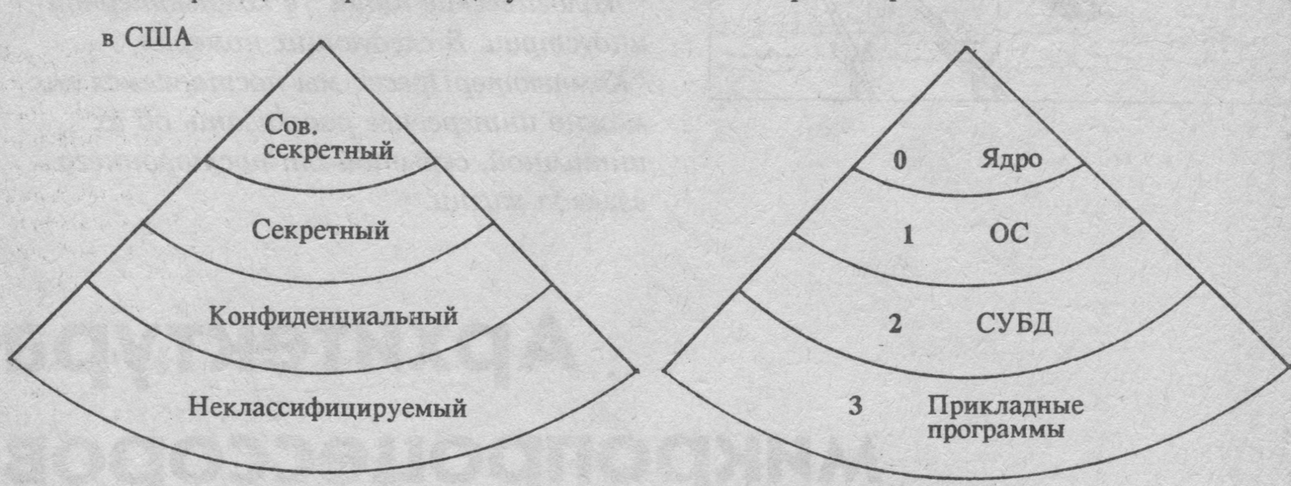


Рис. 3. Байт доступа

7	6 5	0
	DPL	

Рис. 4. Формат селектора

15 … 3	2	1 0
Индекс	TI	RPL

Рис. 5. Вызов через шлюз

0
1 ОС
2
3
Прикладная программа
Шлюз

Рис. 6. Формат шлюза вызова

63	47 … 40		31 … 16	15 … 0
	Доступ		Селектор назначения	Смещение назначения

Равняйсь на Бюрократию или "Тайна второго кольца"

Западные изобретатели — народ практичный и изобретать велосипед не в их привычках — куда дешевле воспользоваться чужим опытом. Так вот, подобную схему приоритетов они взяли не с потолка, а "передрали" прямо с системы защиты документов, которой по сию пору пользуется правительство США. Там определено четыре класса секретности для бумаг, проходящих через руки государственных чиновников: совершенно секретный, секретный, конфиденциальный и неклассифицируемый (рис.1). Но это еще "цветочки" — все население Штатов тоже разбито на четыре класса, которые называются "уровнями благонадежности": совершенно секретный, секретный, конфиденциальный и неблагонадежный. При такой постановке дела совершенно ясно, что забулдыга из Куинз вряд ли получит доступ к стратегическим планам Пентагона. (Вот интересно, сколько уровней благонадежности предусмотрел для советских граждан КГБ и в какую группу он включил сотрудников "КомпьютерПресс"?)

Схема защиты наиболее важных сегментов операционной системы как две капли воды похожа на бюрократическую "луковицу", придуманную американским правительством — те же четыре уровня, только называются они уровнями привилегий, а самый центр с нулевым номером уготован для сверхзащищенного "ядра безопасности". Ядро предназначено для небольшой части ОС, отвечающей за управление памятью и доступом, а также за осуществление собственно защиты. Остальные сегменты операционной системы имеют первый уровень привилегий, на долю же прикладных задач выпадает только третий уровень. Может возникнуть резонный вопрос: чем заполнить второе кольцо безопасности и нужно ли оно вообще? Между прочим, четыре уровня получились, как вы, наверное, догадались не столько из-за слепого подражания, сколько потому, что число 4 кодируется двумя битами. Но "лишний" уровень не долго пустовал — нашлось и для него применение: существуют некоторые программные продукты, например базы данных, которые сами занимаются своей безопасностью, вот им-то второе кольцо и пришлось как раз впору (рис. 2).

Остается пока неясным, каким образом процессор распознает уровень привилегии того или иного сегмента. Здесь нам придется ненадолго вернуться к той главе, где мы знакомились с вычислением виртуального адреса и вспомнить, что назначение сорока битов из 64-х разрядов дескриптора так и осталось для нас загадкой. Оказывается, разряды с 40-го по 47-й любого дескриптора занимает так называемый байт доступа, пятый и шестой биты которого несут информацию о номере кольца защиты (рис. 3). Комбинация этих двух разрядов как раз и является уровнем привилегий дескриптора (DPL). Если производится запись сегмента в ОЗУ, DPL играет роль определителя класса секретности данных, а в том случае, когда сегмент содержит выполняемый код, уровень привилегий превращается в показатель благонадежности программы.

Откуда же берется содержимое DPL? Давайте вспомним структуру селектора (рис. 4). Из шестнадцати его разрядов старшие тринадцать, как вы помните, отводятся под индекс, один служит индикатором таблицы, а вот два младших бита представляют собой запрашиваемый уровень привилегий (RPL). В процессоре предусмотрен специальный механизм поддержки текущего уровня привилегий (CPL), который соответствует степени "благонадежности" (DPL) выполняемой процедуры.

Для плебеев — отдельный вход

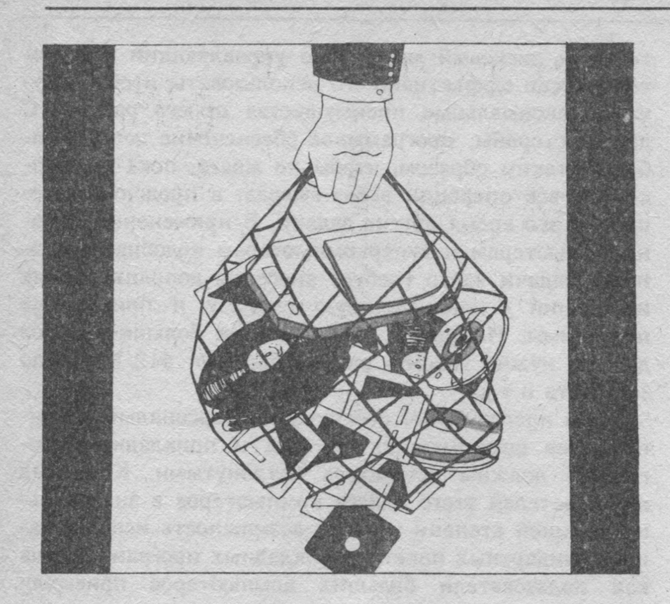
Дальше, вроде бы, все просто: если CPL текущей программы меньше или равен DPL сегмента памяти, то доступ разрешен (вспомним, что нулевой уровень — самый высокий), а если налицо противоположная ситуация, то есть CPL>DPL, доступ запрещен, не так ли? Вы не заметили в этом утверждении ничего странного? Тогда каким же образом прикладной программе достучаться до операционной системы и добиться от нее необходимой поддержки, ведь ее CPL по крайней мере на двойку больше, чем DPL сегментов ОС?

Для того чтобы избежать подобного противоречия, в процессоре реализован механизм вызова более привилегированной процедуры через разрешенные точки входа. Упрощенно это выглядит так: пользовательская программа обращается к специальному дескриптору, определяющему точку входа в процедуру операционной системы (рис. 5). Специальный дескриптор носит название "шлюза вызова", уровень его привилегий достаточно низок для того, чтобы к нему можно было обратиться из любой прикладной программы, а адрес заранее определен. Шлюз, помимо байта доступа, содержит счетчик слов, определяющий количество параметров, передаваемых при вызове, а также селектор и смещение назначения, точно указывающих виртуальный адрес точки входа (рис. 6).

Подобная косвенная передача управления через шлюз вызова имеет свои прелести: можно внести в пользовательскую программу какие угодно изменения или перекомпилировать ее, можно даже установить новую версию операционной системы, если для шлюзов места в GDT (глобальной дескрипторной таблице) остались постоянными, достаточно изменить виртуальный адрес точек входа и можно смело запускать прикладную программу — она не "заблудится".

А теперь — совершенно секретная информация для самых любопытных. Советую читать при плотно закрытых дверях и в полной темноте! В следующем выпуске нашего журнала мы расскажем о том, что делается внутри i860, но это — военная тайна. Тс-с-с...!!!

И. Липкин



В процессе поиска новых мощных средств хранения информации исследователи пришли к выводу, что и старые не так уж плохи, если посмотреть на них с незнакомых ранее позиций. В результате проведенных изысканий появилось самостоятельное направление разработок дисковых магнитных накопителей — дисковые массивы, которые обещают стать новым поколением носителей информации и вытеснить в дальнейшем своих прародителей.

ДИСКОВЫЕ МАССИВЫ

Технический прием объединения магнитных дисков в дисковые массивы (disk arrays) становится основным в технологии создания современных мощных накопителей информации. За последние несколько лет разработчики аппаратных средств испробовали множество способов повышения скорости обмена информацией, записанной на магнитных носителях. При помощи дисковых массивов удалось добиться повышения надежности и увеличения скорости передачи информации, а также улучшить характеристики ввода-вывода. Но при всех достоинствах дисковых массивов имеются значительные трудности в разработке контроллеров, программного обеспечения и систем обработки ошибок.

Дисковые массивы воплощают простую идею о том, что несколько дисков, объединенных вместе, могут передавать информацию быстрее, чем один диск.

Самая популярная конфигурация, с четырьмя дисками данных и одним диском контроля четности, перемещает данные почти в четыре раза быстрее, чем одиночный диск. Например, система, использующая четыре SCSI (Small Computer System Interface) диска со скоростью передачи данных 3 Мбайт/сек каждый, может перемещать данные со скоростью 10 Мбайт/сек. В сравнении с этим, недавно появившиеся одиночные диски SCSI-2 обеспечивают скорость обмена 10 Мбайт/сек только при помощи весьма напряженного использования кэш-памяти. Более того, если разработчики дисковых массивов начнут использовать SCSI-2 диски, четырехдисковые массивы смогут обеспечить скорость передачи информации до 40 Мбайт/сек, хотя здесь возможно некоторое снижение скорости за счет влияния других факторов.

В своем большинстве создатели систем дисковых массивов утверждают, что это не слишком трудная задача. Не так сложно получить систему, которая позволит вам преодолеть барьер скорости передачи данных в 30-40 Мбайт/сек. Однако, прежде чем приступить к созданию дисковых массивов, разработчики должны потратить массу времени на проведение серьезных исследований.

Главная проблема заключается в выборе подходящего класса дискового массива. Различные классы массивов отличаются скоростью доступа к информации, скоростью передачи информации и требованиями к ее надежности. При оптимизации массива под конкретную задачу инженер в первую очередь должен расставить приоритеты между всеми составляющими ее аспектами.

Классы дисковых массивов

В одном уважаемом издании Калифорнийского Университета дисковые массивы называются "резервирующими массивами недорогих дисков" (Redundunt Arrays of Inexpensive Disks — RAID) и делятся на пять

классов. Исследование, первоначально выполненное для выяснения преимуществ дисковых массивов из 5,25- 3,5- и даже 8-дюймовых дисков перед большими дисками, теперь служит руководством для определения типа стоящей перед разработчиком задачи: получение дополнительного объема, максимальной скорости передачи информации или высоких характеристик ввода-вывода.

Во всех пяти классах оцениваются преимущества дисковых массивов по сравнению с одиночными дисками в скорости передачи информации, высоких характеристиках ввода-вывода и надежности передаваемой информации, но различные классы ориентированы на разные задачи. В большинстве обзоров рассматривается лишь три класса дисковых массивов — первый, третий и пятый — как наиболее распространенные.

Дисковые массивы первого класса имеют полный дубликат каждого диска, обеспечивая максимальную надежность хранения информации, но также и наивысшую стоимость всей системы. Третий класс поддерживает максимальную скорость передачи данных, которая нужна при решении графических и других задач с большими файлами. Пятый класс дисковых массивов ориентирован на напряженную работу с дисками и обеспечивает максимальную скорость доступа к информации.

Необходимость удовлетворения множеству условий приводит к тому, что процесс выбора конфигурации дискового массива может стать крайне сложным. Разработчики компьютерной системы должны четко представлять себе, как она будет использоваться конечными потребителями, и идти навстречу нуждам пользователей. Некоторые изготовители высокопроизводительных компьютеров покупают уже готовые дисковые массивы у других фирм, но многие все же предпочитают разрабатывать свои собственные, по большей части из-за сложности согласования характеристик массива с требованиями всей компьютерной системы.

После того, как выбран класс дискового массива, начинается его разработка, в процессе которой приходится решать значительно более сложные задачи, чем при создании одиночных дисковых систем. Разработчики должны учитывать многие особенности дисковых массивов, как например, необходимость использования блоков большего размера, отслеживания ошибок на отдельном диске, в то время как идет чтение с других и т.п.

Необходимо принимать во внимание особенности архитектуры компьютера, а чтобы получить высокие характеристики ввода-вывода, нужно оптимизировать размеры передаваемых файлов. Помимо этого, нужно должным образом подобрать характеристики используемых дисководов, чтобы достичь максимальной скорости передачи данных.

Разработчики дисковой системы должны позаботиться также и о сбалансированности ее производительности. Дисковые массивы способны передавать большие объемы данных, чем их может пропустить системная шина. Если у вас есть высокопроизводительный дисковый массив, то управляющий компьютер должен эффективно его использовать, чтобы получить максимальные преимущества от его работы. С другой стороны, программное обеспечение должно работать таким образом, чтобы не ждать, пока диск закончит все операции ввода-вывода, а продолжать решать в это время другие задачи. В применении к миникомпьютерам и суперкомпьютерам подобная постановка задачи часто требует внесения дополнительных изменений в операционную систему и прикладные программы. Например, для передачи больших блоков данных нужно удвоить размер блока от 512 Кбайт до 1 Мбайта и т.д.

Но в программном обеспечении персональных компьютеров операционная система и прикладные программы должны оставаться нетронутыми. Конечных пользователей этого класса компьютеров в значительно большей степени заботит возможность использования стандартных пакетов прикладных программ, тогда как пользователи больших компьютеров привыкли постоянно изменять свои программы для получения наивысшей производительности. Это означает, что разработчики дисковых массивов для персональных компьютеров должны потратить гораздо больше усилий на то, чтобы преодолеть ограничения, налагаемые имеющимся программным обеспечением.

Стандартные операционные системы персональных компьютеров — DOS, OS/2, Novell, PC Unix и другие — не обязательно должны знать, как работать с дисками объема 4 или 5 Гбайт. Эти операционные системы имеют некоторые свои особенности и ограничения, которые должны быть обойдены на аппаратном уровне, с тем чтобы программное обеспечение оставалось нетронутым, а дисковый массив мог работать в полную силу. Для этих целей используются специальные драйверы.

Однако разработчики дисковых массивов говорят, что в скором времени даже для персональных компьютеров придется вносить определенные изменения в их операционные системы для более эффективной работы с дисковыми массивами. Многие операционные системы для адресации дисков используют 32-битные слова, и, следовательно, они ограничены объемом 2 Гбайта. Если вы объединяете в массив пять дисков с объемом более 400 Мбайт каждый, то у вас появляется область дискового пространства, которую операционная система не может адресовать. И этот недостаток тоже должен быть как-то устранен.

Управление ошибками

Поскольку дисковые массивы требуют лишь незначительных изменений в программном обеспечении, то для поддержки их высокой производительности необходим очень интеллектуальный контроллер. В режиме записи контроллер делит файл на четыре или более частей, каждая из которых передается на свой дисковод. Когда нужно прочитать данные, контроллер собирает их воедино и пересылает запрашивавшему ус-

тройству, которому нет дела до того, что данные были разделены.

Для увеличения производительности контроллер должен так организовать процессы чтения и записи, чтобы какой-либо из дисководов не бездействовал, пока все другие заняты делом, а также чтобы головки дисков не совершали излишних перемещений.

Контроллер должен просматривать все запросы на обращения к дисковому массиву. Затем он должен так распределить очередность выполнения этих запросов, чтобы загруженность дисков была равномерной, а их работа максимально эффективной.

Поскольку на каждом диске содержится лишь часть данных, контроллер должен уметь быстро собирать эти части воедино, не забывая при этом следить за новыми запросами и подготавливать их выполнение. Неудивительно, что одной из ключевых задач контроллера дискового массива является эффективная и надежная обработка ошибок.

Необходимо иметь полную гарантию того, что все возможные ошибки будут обнаружены. В устройствах с несколькими дисками, объединенными в общий массив, существует вероятность появления таких ситуаций, которые никогда не возникали прежде в однодисковых системах. Причем дисковые массивы могут оказать в таких случаях усиливающее воздействие и вызвать лавину новых проблем.

При разработке дисковых массивов делается попытка, насколько это возможно, эмулировать одиночный диск в части системных обращений к нему. Это заставляет в значительной степени усложнять контроллер. Главная же задача — обеспечить высокую достоверность передаваемой информации. Безусловно, все ошибки должны быть надежно устранены.

Контроллеры должны отслеживать поврежденные места на дисках, отказы любого из дисководов, ошибки плохого электрического контакта, а также обычные битовые ошибки записи и чтения. Основная проблема состоит в том, чтобы уведомить запрашивающее устройство о возникшей неприятности. Большинство операционных систем предназначены для работы с одиночными дисками и не понимают тех ситуаций, когда один из дисков отказывает, но дисковая система в целом продолжает оставаться работоспособной.

Другая сложная задача состоит в необходимости поиска на дисках маленьких отрезков данных, которые не уместились в кэш-память при считывании концов файлов. При этом контроллер должен определить, где находится искомый блок, затем выяснить, свободен ли соответствующий канал и, наконец, считать данные с нужного диска, не затрагивая остальных.

Серьезной задачей при поиске ошибок является необходимость осуществления этого поиска с минимальными затратами ресурсов. Использование алгоритмов более сложных, чем это необходимо в конкретной ситуации, требует дополнительного времени на обработку ошибок.

Пару лет назад разработчики дисковых массивов пришли к выводу, что нет необходимости производить одиночную или двойную коррекцию ошибок в оперативной памяти компьютера, поскольку в случае отказа одного из дисков восстановить данные не представляет большого труда. Контроль четности легко позволяет справиться с этим.

Самый простой способ повышения надежности хранения информации на нескольких дисках — это добавление диска контроля четности. Когда теряются данные на одном из дисков, всегда есть достаточно информации на диске контроля четности и остальных дисках для надежного восстановления потерянных данных.

Задача восстановления данных должна решаться совместными усилиями разработчиков дисковых массивов и прочих системных средств. Некоторые конечные пользователи хотят иметь возможность продолжать работу одновременно с восстановлением потерянной информации на замененном диске. Это довольно сложная проблема, поскольку восстановление данных связано с перемещением сотен мегабайт с остальных дисков, каждый из которых содержит лишь часть необходимой информации. Тысячи актов чтения и записи не только предельно загружают все диски массива, но и требуют максимального напряжения от контроллера, который должен распределять время между процессом восстановления информации и обычной работой.

Простейший способ восстановления потерянных файлов заключается во временной остановке системы для загрузки информации на замененный диск. Но для пользователей, которые не могут позволить себе таких задержек, замена диска все равно приведет к временному снижению производительности, поскольку контроллер должен одновременно с обслуживанием системных запросов восстанавливать потерянные файлы.

Другая задача разработки состоит в том, чтобы решить, как быстро дисковый массив должен восстанавливать потерянную информацию. Добавление лишних микропроцессоров в контроллер уменьшит время восстановления, но и увеличит стоимость всей системы. Помимо этого, сложности программирования дополнительных процессоров и их синхронизации также прибавят разработчикам лишнюю головную боль.

Многое в обнаружении и обработке ошибок зависит от изготовителей контроллеров и прочих подсистем дисковых массивов, которые прилагают большие усилия для достижения гарантии надежной работы всей дисковой системы. Однако компоновщики конечных пользовательских систем также должны осуществлять подобное тестирование для получения уверенности в четкой работе массивов в конкретном аппаратном окружении.

Существуют тысячи ситуаций, при которых информация на дисках может быть потеряна, и тысячи способов устранения связанных с этим ошибок и восстановления данных. Разработчики предлагают множество аппаратных решений, гарантирующих сохранность данных без их резервного копирова-

2 Зак. 2260

"Мы хотели осуществить качественно новый скачок, не вкладывая дополнительных средств в повышение производительности отдельных дисков", — сказал Майк Перез (Mike Perez), технический директор Compaq.

В то время как фирма Compaq определяла техническую политику дисковых массивов для персональных компьютеров, Tandem Computers разрабатывала ее для систем непрерывного пользования, купив линию по производству дисковых массивов, а компания NCR недавно начала создание массивов для больших систем.

Те, кто обычно ждет каких-либо шагов со стороны IBM, не будут слишком долго томиться в неопределенности: компания уже подписала соглашение с фирмой Maximum Strategy (San Jose, Calif.), производителем дисковых подсистем. Совместно они будут разрабатывать массивы, подключаемые к системам IBM через канал высокопроизводительного параллельного интерфейса (HiPPI — High Performance Parallel Interface), который становится универсальным для индустрии суперкомпьютеров.

Несмотря на то, что к дисковым массивам проявляется большой интерес, некоторые изготовители контроллеров считают, что этот рынок не так уж велик. Ни компании Xylogics (Burlington, Mass), ни Interphase (Dallas) — два главных производителя дисковых контроллеров — не собираются выпускать контроллеры дисковых массивов.

"Все в первую очередь заинтересованы в прибылях, но всякий интерес тут же пропадает, как только речь заходит об увеличении себестоимости продукции в два раза. Производители не хотят увеличивать себестоимость более чем на 10 процентов. Как только кто-либо достигнет подобных результатов, рынок будет готов принять дисковые массивы", — говорит Эрнст Годси (Ernest Godsey), директор по маркетингу компании Interphase.

Но, невзирая на то, что рынок контроллеров дисковых массивов может быть не слишком большим, производители как самих контроллеров, так и дисковых подсистем твердо намерены продолжать совершенствовать эту технологию.

Современный рынок дисковых средств совершенно определенно наметил дисковые массивы в качестве своих главных перспектив на ближайшее время. На сегодняшний день не осталось ни одного серьезного производителя вычислительных систем, который бы не проводил исследования по включению дисковых массивов в состав своих будущих разработок.

А. Синев

По материалам:
Electronic Engineering Times, November 5, 1990.

Плетью обуха не перешибешь.

Несмотря на отсутствие какой бы то ни было правовой защиты от нелегального копирования и отсутствие каких бы то ни было штрафных санкций за использование нелегального программного обеспечения, корпорация Microsoft намерена поставлять вторую редакцию локализованного интегрированного пакета WORKS 2.0 без программной защиты.

Для справки, первая редакция пакета поставлялась с системой защиты, позволяющей использовать рабочую копию пакета не более чем на одном компьютере одновременно. При переносе рабочей копии на другой компьютер существующая копия удалялась.

Представители фирмы утверждают, что отказ от программной защиты вызван желанием устранить дискомфорт потребителя, вызванный возможностью утраты пакета при сбое жесткого диска, повреждении оригинальной дискеты или заражении. Кроме того, они признались, что установка защиты на первую редакцию пакета была ошибкой.

Во второй редакции исправлен ряд недостатков первой, в том числе некорректная работа режима рисования псевдографики.

Локализованный интегрированный пакет WORKS завоевывает все большую популярность и достаточно быстро распространяется по стране, но не приносит ни копейки прибыли фирме-разработчику. Автором одной из вскрытых копий злые языки называют известного программиста А.Чижова (одного из авторов резидентного руссификатора РусскоеСлово, защищенного программой датской фирмы LinkComputer), если это верно, то остается надеяться, что он сделал это из спортивного интереса.

Вызывает большое сомнение, что продажа на советском рынке незащищенного пакета, в надежде на честную игру советских пользователей, принесет фирме дивиденды и следовательно желание локализовать новые программные продукты.

Следует обратить внимание на то обстоятельство, что корпорация Microsoft прекратила поставки локализованной версии ДОС 4.01 и в СССР торгует только лицензиями на производство.

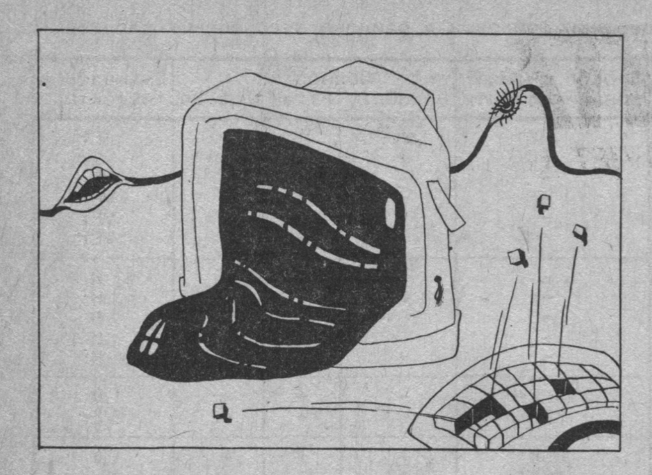
В стране, где обману учат сказки, а воровство почитается, есть только один путь для нормальной коммерческой деятельности в области программного обеспечения — это снижение цены до такого уровня, когда выгоднее будет купить, чем пользоваться нелегальными копиями.

В цивилизованных странах купить, это значит:
получить документацию;
получить запечатанный пакет дистрибутивных дисков;
получать необходимую информацию и консультации;
присутствовать на выставках и семинарах;
заменять старые версии на новые за треть или четверть цены;
приобретать новые продукты по сниженным ценам.

Много ли в СССР организаций — распространителей программного обеспечения — предоставляют подобный сервис? Большинство направляет свои доходы по дилерскому проценту куда угодно, но не на поддержку пользователей и развитие дилерской службы.

Если корпорации Microsoft удастся открыть производство программных продуктов в СССР, реализовывать их за рубли (за приемлемую цену) и организовать сеть поддержки, тогда покупать будет выгодно, а нынешняя цена пакета MS-WORKS 2.0 — 350 долл., не оставляет никаких иллюзий на счет путей дальнейшего распространения второй редакции этого пакета.

Newsbytes News Network



КОМПЬЮТЕРНЫЕ ВИРУСЫ: ПРЕДВАРИТЕЛЬНЫЕ СООБРАЖЕНИЯ

Мы и обернуться не успеем, а уже навалят больше прежнего...

А. и Б.Стругацкие "Град обреченный"

Стоит мне что-нибудь проглотить, как тут же происходит что-нибудь интересное. Посмотрим, что будет на этот раз!

Л.Кэролл "Алиса в стране чудес"

По поводу термина

Первые исследования саморазмножающихся искусственных конструкций проводились в середине нынешнего столетия: в работах фон Неймана, Винера и др. дано определение и проведен математический анализ конечных автоматов, в том числе самовоспроизводящихся. Термин "компьютерный вирус" появился позднее — официально считается, что его впервые употребил сотрудник Лехайского университета (США) Фред Коэн в 1984 году на 7-й конференции по безопасности информации, проходившей в США. С тех пор прошло немало времени, острота проблемы вирусов нисколько не снизилась, однако строгого определения, что же такое компьютерный вирус, так и не дано, несмотря на неоднократные попытки дать такое определение. Поэтому я наберусь смелости и попытаюсь сформулировать собственное определение.

Компьютерным вирусом называется программа, которая может создавать свои копии (не обязательно полностью совпадающие с оригиналом) и внедрять их в файлы, системные области компьютера, сети и так далее. При этом копии сохраняют способность дальнейшего распространения.

По поводу самих вирусов и их авторов

Сам я не написал ни одного вируса, не знаком ни с одним из авторов вирусов, и, следовательно, мои соображения по этому поводу могут быть лишь чисто теоретическими.

Так кто же пишет вирусы? На мой взгляд, основную массу вирусов создают люди, которые только что изучили язык ассемблера, хотят попробовать свои силы, но не могут найти для них более достойного применения. Из под пера подобных "умельцев" часто выходят либо модификации "классических" вирусов, либо вирусы крайне примитивные и с большим числом ошибок (такие вирусы я называю "студенческими").

Самую опасную группу составляют "профессиональные" вирусы. Эти тщательно продуманные и отлаженные программы создаются профессиональными, нередко очень талантливыми программистами.Такие вирусы зачастую используют достаточно оригинальные алгоритмы, недокументированные и мало кому известные способы проникновения в системные области. Часто "профессиональные" вирусы выполнены по технологии "стелс". Для меня остаются загадкой причины, которые могут заставить человека направить свои немалые способности на такую бессмысленную работу. Не исключено, что некоторые из таких людей — неудачники, не сумевшие более эффективно воспользоваться своими знаниями.

Умрет ли MS-DOS от вирусной болезни?

По моим сведениям, в развитых странах проблема компьютерных вирусов не является столь острой, так как там в значительно меньшей степени распространено пиратское копирование программного обеспечения, гораздо более активно действуют фирмы, специа-

лизирующиеся на производстве антивирусной продукции, много разнообразной литературы, посвященной данному вопросу. В результате большинство пользователей знают, что такое вирус и какие могут быть последствия от его "визита", готовы к борьбе с вирусом, но "живьем" ни одного вируса не видели уже очень давно.

В нашей же стране все обстоит гораздо более серьезно. Основным источником приобретения новых программных продуктов у нас является их бесконтрольное копирование; персональные компьютеры очень часто используются в "многоперсональном" режиме. Для распространения вируса это самая благодатная среда. В результате в учебных институтах, в организациях (иногда достаточно серьезных) факт заражения вирусом вполне обычное явление, что-то вроде насморка. Такое "привыкание" может кончиться плачевно при заражении каким-либо опасным вирусом. Хотя, если на машине нет ничего, кроме DOS и большого числа игр, то вируса можно не бояться — невелика потеря (кстати, ситуация очень распространенная).

Таблица 1. Свойства и характеристики файловых и файлово-загрузочных вирусов.

Имя	Резидентность	Метод заража	Способ инфицирования памяти	Обработка ReadOnly	FTime	Int 24	Вызываемые эффекты
Amstrad-277	–	FIND	–	–	–	–	Er
Amstrad-299	–	FIND	–	–	–	–	VEr
Amstrad-345	–	FIND	–	–	–	–	VEr
Amstrad-740	–	FIND	–	–	–	–	VEr
Amstrad-847	–	FIND	–	–	–	–	VEr
Amstrad-852	–	FIND	–	–	–	–	VEr
Anti-Pas-400	–	FIND	–	+	+	–	H
Anti-Pas-440	–	FIND	–	+	+	–	H
Anti-Pas-480	–	FIND	–				H
Anti-Pas-529	–	FIND	–				H
Anti-Pas-605	–	FIND	–	–	–	+	H
April 1st COM	+	INT	KEEP	–	+	+	V
April 1st EXE	+	INT	KEEP	–	+	+	VEr
Cascade-1701	+	INT	MCB*	+	+	–	V
Cascade-1704	+	INT	MCB*	+	+	–	V
Eddie-651	+	INT	MCB	+	+*	+	–
Eddie-1800	+	INT	MCB	+	+	+	H
Eddie-2000	+	INT	MCB	+	+*	+	H
Eddie-2100	+	INT	MCB	+	+*	+	H
Jerusalem	+	INT	KEEP	+	+	+	HEr
Fumanchu-2080	+	INT	KEEP	+	+	+	VEr
Kylie-2272	+	INT	KEEP	+	–	+	MEr
SunDay-1631	+	INT	KEEP	+	+	+	HEr
Taiwan-2900	+	INT	KEEP	+	–	+	HMEr
Hymn-1865	+	INT	MCB	+	+	–	VMH
Hymn-1962	+	INT	MCB	+	+	+	VMH
Hymn-2144	+	INT	MCB	+	+	+	VMH
Murphy-1277	+	INT	MCB	–	+	+	M
Murphy-1521	+	INT	MCB	–	+	+	V
Phoenix	+	INT*	MCB	+	+	+	H
Evil	+	INT*	MCB	+	+	+	H
Proud	+	INT*	MCB	+	+	+	H
Saratoga-632	+	INT	MCB	–	–	–	–
Saratoga-642	+	INT	MCB	–	–	–	Ch
Saratoga-656	+	INT	MCB	–	–	–	Ch
SVC-1689	+	INT	MCB	+	+	+	–
SVC-1740	+	INT	MCB	+	+	+	–
SVC-3103	+	INT	MCB	+	+	+	–
Tiny-133	+	INT	*	–	–	–	–
Tiny-134	+	INT	*	–	–	–	–
Tiny-138	+	INT	*	–	–	–	–
Tiny-143	+	INT	*	–	–	–	–
Tiny-154	+	INT	*	–	–	–	–
Tiny-156	+	INT	*	–	–	–	–
Tiny-158	+	INT	*	–	–	–	–
Tiny-159	+	INT	*	–	–	–	–
Tiny-160	+	INT	*	–	–	–	–
Tiny-167	+	INT	*	–	–	–	–
Tiny-198	+	INT	*	–	–	–	–
V-1024-a	+	INT	MCB	+	+	+	V
V-1024-b	+	INT	MCB	+	+	+	V
V-600	+	INT	MCB	+	+	+	–
Voronezh-1600	+	INT	MCB	+	+	+	Er

Следует отметить быстрый рост числа компьютерных вирусов советского производства (по моим данным, общее число на 1989г. — 1, на август 1990г. — 3 или 4, сентябрь — около 6, октябрь — около 15, ноябрь — около 20, затем я сбился со счету). Это естественно — ведь когда платят лишь за присутствие на работе, есть возможность заниматься всем, чем угодно (попробуйте возразить, что, мол, это очень редкая ситуация), а ничем полезным или прибыльным заниматься не хочется, то от нечего делать можно попробовать запрограммировать вирус (тем более, что в нашей стране это пока не преследуется уголовно). К тому же программирование все больше и больше превращается из искусства в ремесло, а среди программистов все реже встречаются "ин-

теллектуалы“ и все чаще ремесленники-“таперы“, что не может не сказаться на “индексе порядочности“ среди представителей этой профессии. В результате компьютерное хулиганство процветает (именно хулиганство, поскольку создавать вирусы и мочиться в лифте — поступки одного порядка).

Таблица 1. Свойства и характеристики файловых и файлово-загрузочных вирусов.
(Продолжение)

Vienna-348	–	FIND	–	+	+*	–	H
Vienna-353	–	FIND	–	+	+*	–	H
Vienna-367	–	FIND	–	+	+*	–	H
Vienna-435	–	FIND	–	+	+*	–	H
Vienna-507	–	FIND	–	+	+*	–	–
Vienna-534	–	FIND	–	+	+*	–	–
Vienna-623	–	FIND	–	+	+*	+	H
Vienna-627	–	FIND	–	+	+*	–	–
Vienna-644	–	FIND	–	+	*	–	–
Vienna-648	–	FIND	–	+	+*	–	H
Vienna-757	–	FIND	–	+	+*	+	V
Vienna-776	–	FIND	–	+	+*	+	V
Big Piter	– *	FIND	–	+	+*	–	V
VACSINA-04	+	INT	MCB	+	–	+	M
VACSINA-05	+	INT	MCB	+	–	+	M
VACSINA-06	+	INT	MCB	+	–	+	M
VACSINA-10	+	INT	MCB	+	+	+	M
Yankee - 17	+	INT	MCB	+	+	+	M
Yankee - 18	+	INT*	MCB	+	+	+	M
Yankee - 19	+	INT	MCB	+	+	+	M
Yankee - 21	+	INT	MCB	+	+	+	M
Yankee - 22	+	INT	MCB	+	+	+	M
Yankee - 26	+	INT	MCB	+	+	+	M
Yankee - 27	+	INT	MCB	+	+	+	M
Yankee - 29	+	INT	MCB	+	+	+	M
Yankee - 2A	+	INT	MCB	+	+	+	M
Yankee - 2C	+	INT	MCB	+	+	+	M
Yankee - 2D	+	INT	MCB	+	+	+	M
Yankee - 2E	+	INT	MCB	+	+	+	M
Yankee'-1905	+	INT	MCB	+	+	+	M
Yankee'-1049	+	INT	MCB	+	+	+	–
YanShort-1624	–	FIND	–	–	+	–	*
YanShort-1961	–	FIND	–	–	+	–	M
512 (вер a-f)	+	INT	Buffer	–	+*	+	–
Alabama	+	INFI	MCB	+	+*	+	V
ATTENTION	+	INT	MCB	+*	–	+*	Er
Bebe	– *	FIND	*	–	–	–	V
Datacrime	–	FIND	–	+	+	–	VH
Joker-01	+	INT	MCB	+	+	+	V
Kemerovo	–	FIND	–	–	–	–	*
Kuku-448	–	FIND	–	+	*	–	HV
Lehigh	+	INT	MCB	–	–	–	H
Liberty	+	INT	MCB	–	–	+	H
LoveChild	+	INT	IntTable	–	–	+	H
Loz-1023	+	INT	MCB	*	–	+	HV
Minsk-1075	+	INT	MCB	+	–	+	HEr
Mirror	+ *	INT	?	–	–	–	VH
MGTU-273	–	FIND	–	–	–	–	–
MLTI-830	+	INT	MCB	+	–	–	H
Peterburg	+	INT	KEEP	+	+	+	–
RC-492	+	INT	IntTable	–	–	–	H
Small-144	+	INT	IntTable	–	–	–	–
Sylvia	–	FIND	–	–	–	+	–
VCOMM-637	–	FIND	–	+	+	–	*
V-516	+	INT	Buffer	+	+	+	Ch
V-707	+	INT	MCB*	+	+	–	V
V-905	+	INT	MCB	+	+	+	V*
Wish-1024	+	INT	MCB	+	–	+	H
4096	+	INT	MCB*	+	+*	+	VH

КЛАССИФИКАЦИЯ КОМПЬЮТЕРНЫХ ВИРУСОВ

Вирусы можно разделить на классы по следующим признакам:

— по среде обитания вируса;

— по способу заражения среды обитания;

— по деструктивным возможностям.

По среде обитания различают вирусы сетевые, файловые и загрузочные. Сетевые вирусы распространяются по компьютерной сети, файловые внедряются в выполняемые файлы, загрузочные — в загрузочный сектор диска (Boot-сектор) или в сектор, содержащий системный загрузчик винчестера (Master Boot Record). Существуют сочетания — например, файлово-загрузочные вирусы, заражающие и файлы и загрузочные сектора дисков. Кроме того, по сети могут распространяться вирусы любых типов.

Способы заражения делятся на резидентный и нерезидентный. Резидентный вирус при инфицировании компьютера оставляет в оперативной памяти свою резидентную часть, которая затем перехватывает обращение операционной системы к объектам заражения и внедряется в них. Резидентные вирусы находятся в памяти и остаются активными вплоть до выключения или перезагрузки компьютера. Нерезидентные вирусы не заражают память компьютера и являются активными ограниченное время. Некоторые вирусы оставляют в оперативной памяти небольшие резидентные программы, которые не распространяют вирус. Такие вирусы считаются нерезидентными.

Таблица 1. Свойства и характеристики файловых и файлово-загрузочных вирусов. (продолжение).

Имя	Длина (L)	Заражаемые файлы	Место зараж	Диапазон длин заражаемых файлов	Увеличение длин файлов '$' – доп. до параграфа
Amstrad-277	277 (115h)	.C	e	не проверяет	L, ++
Amstrad-299	299 (12Bh)	.C	e	не проверяет	L, ++
Amstrad-345	345 (159h)	.C	e	не проверяет	L
Amstrad-740	740 (2E4h)	.C	e	не проверяет	L
Amstrad-847	847 (34Fh)	.C	e	не проверяет	L, ++
Amstrad-852	852 (354h)	.C	e	не проверяет	L, ++
Anti-Pas-400	400 (190h)	.C	e	2048 (800h) <C< 64000 (FA00h)	L*
Anti-Pas-440	440 (1B8h)	.C	e	2048 (800h) <C< 64000 (FA00h)	L*
Anti-Pas-480	480 (1E0h)	.C	e	2048 (800h) <C< 64000 (FA00h)	L*
Anti-Pas-529	529 (211h)	.C	s	529 (211h) <C< (FDEEh)	L*
Anti-Pas-605	605 (25Dh)	.C	s	605 (25Dh) <C< (FDA2h)	L*
April 1st COM	897 (381h)	.C	s*	не проверяет	L
April 1st EXE	1488 (5D0h)	.E	i	не проверяет	L
Cascade-1701	1701 (6A5h)	C	e	не проверяет	L
Cascade-1704	1704 (6A8h)	C	e	не проверяет	L
Eddie-651	651 (28Bh)	CEO	e	651 (28Bh) <C< (FB75h), 651 (28Bh) <E	L
Eddie-1800	1800 (708h)	CEO	e	1775 (6EFh) <C< 63149 (F6ADh), 1775 (6EFh) <E	L
Eddie-2000	2000 (7D0h)	CEO	e	1959 (7A7h) <C< 62947 (F5E3h), 1959 (7A7h) <E	L
Eddie-2100	2100 (834h)	CEO	e	(809h) <C< (F580h), (809h) <E	L
Jerusalem	1808 (710h)	.C.E *	CsEe	не проверяет	C– L+4, E– L+$, E – ++
SunDay-1631	1631 (65Fh)	.C.E	CsEe	не проверяет	C– L+4, E– L+4+$
Kylie-2272	2272 (8E0h)	.C.E	CsEe	не проверяет	L+?
Fumanchu-2080	2080 (820h)	.C.E	CsEe	не проверяет	C– L, E– L+$
Taiwan-2900	2900 (B54h)	.C.E	CsEe	не проверяет	C– L, E– L+$
Hymn-1865	1865 (749h)	CE	e	(B00h) <C< (F66Bh), (B00h) < E	L
Hymn-1962	1962 (7AAh)	CE	e	(B00h) <C< (F60Ah), (B00h) < E	L
Hymn-2144	2144 (860h)	CE	e	(D00h) <C< (F554h), (D00h) < E	L
Murphy-1277	1277 (4FDh)	(CE)&(.C.E)	e	1277 (4FDh) <C< 64226 (FAE2h), 1277 (4FDh) <E	C – L, E – $+L
Murphy-1521	1521 (5F1h)	(CE)&(.C.E)	e	1521 (5F1h) <C< (F9EEh), 1521 (5F1h) <E	C – L, E – $+L
Phoenix	1704 (6A8h)	CE	EeCi*	1960 (7A8H) < C < 64K (FFFFh), E – не проверяет	C – L или 0, E – 132
Evil	1701 (6A5h)	CE	EeCi*	1956 (7A4H) < C < 64K (FFFFh), E – не проверяет	C – L или 0, E – 132
Proud	1102 (516h)	C	Ci*	< C <	C – L или 0
Saratoga-632	632 (278h)	.E	e	не проверяет	L+$
Saratoga-642	642 (282h)	.E	e	не проверяет	L+$
Saratoga-656	656 (290h)	.E	e	не проверяет	L+$
SVC-1689	1689 (699h)	CE	e	1024 (400h) <C< (F943h), E – не проверяет	L
SVC-1740	1740 (6CCh)	CE	e	1024 (400h) <C< (F910h), E – не проверяет	L
SVC-3103	3103 (C1Fh)	CE	e	3103 (C1Fh) <C< (EDE1h), E – не проверяет	L

Таблица 1. Свойства и характеристики файловых и файлово-загрузочных вирусов. (продолжение).

Имя	Длина (L)	Заражаемые файлы	Место зараж	Диапазон длин заражаемых файлов	Увеличение длин файлов '$' - доп. до параграфа
Tiny-133	133 (85h)	C	e	не проверяет	L
Tiny-134	134 (86h)	C	e	не проверяет	L
Tiny-138	138 (8Ah)	C	e	не проверяет	L
Tiny-143	143 (8Fh)	C	e	не проверяет	L
Tiny-154	154 (9Ah)	C	e	не проверяет	L
Tiny-158	158 (9Eh)	C	e	не проверяет	L
Tiny-159	159 (9Fh)	C	e	не проверяет	L
Tiny-160	160 (A0h)	C	e	не проверяет	L
Tiny-167	167 (A7h)	C	e	не проверяет	L
Tiny-198	198 (C6h)	C	e	не проверяет	L
V-1024-a	1024 (400h)	CEO	e	1024 (400h) <C< 64000 (FA00h), 1024 (400h) < E	L
V-1024-b	1024 (400h)	CEO	e	1024 (400h) <C< 64000 (FA00h), 1024 (400h) < E	L
V-600	600 (258h)	C	s*	600 (258h) <C< 60000 (EA60h)	L
Voronezh-1600	1600 (640h)	CE	CsEe	1600 (640h) <C< 61000 (EE48h), E - любой длины	L
Vienna-348	348 (15Ch)	.C	e	10 (0Ah) < C < 64000 (FA00h)	L
Vienna-353	353 (161h)	.C	e	10 (0Ah) < C < 64000 (FA00h)	L
Vienna-367	367 (16Fh)	.C	e	10 (0Ah) < C < 64000 (FA00h)	L
Vienna-435	435 (1B3h)	.C	e	10 (0Ah) < C < 64000 (FA00h)	L
Vienna-507	507 (1FBh)	.C	e	256 (100h) < C < 64000 (FA00h)	L
Vienna-534	534 (216h)	.C	e	256 (100h) < C < 64000 (FA00h)	L
Vienna-623	623 (26Fh)	.C	e	10 (0Ah) < C < 64000 (FA00h)	L
Vienna-627	627 (273h)	.C	e	10 (0Ah) < C < 64000 (FA00h)	L
Vienna-644	644 (284h)	.C	e	1280 (500h) < C < 64000 (FA00h)	L
Vienna-648	648 (288h)	.C	e	10 (0Ah) < C < 64000 (FA00h)	L
Vienna-757	757 (2F5h)	.C	e	(31Ah) < C < (FCF9h)	L, ++
Vienna-776	776 (308h)	.C	e	(32Dh) < C < (CE6h)	L
Big Piter	23693 (5C8Dh)	.C	e	10 (0Ah) < C < 64000 (FA00h)	L
VACSINA-04	1212 (4BCh)	CE *	e	1212 (4BCh) <C< 62855 (F587h), E< 64947 (FDB3h)	C - $+L, E - 132+$+L
VACSINA-05	1206 (4B6h)	CE	e	1206 (4B6h) <C< 62867 (F593h), E< 64947 (FDB3h)	C - $+L, E - 132+$+L
VACSINA-06	1269 (4F5h)	CE	e	1269 (4F5h) <C< (FDB3h), E< (F515h)	C - $+L, E - 132+$+L
VACSINA-10	1239 (53Bh)	CE	e	1339 (53Bh) <C< 62601 (F489h), E< 64947 (FDB3h)	C - $+L, E - 132+$+L
Yankee - 17	1753 (6D9h)	CE	e	32 (20h) < CE < (F058h)	C - L, E - $+L
Yankee - 18	1760 (6E0h)	CE	e	32 (20h) < CE< 61559 (F077h)	C - L, E - $+L
Yankee - 19	1805 (70Dh)	CE	e	32 (20h) < CE < 61469 (F01Dh)	C - L, E - $+L
Yankee - 21	2680 (A78h)	CE	e	32 (20h) < CE < 59447 (E837h)	C - L, E - $+L
Yankee - 22	2568 (A08h)	CE	e	32 (20h) < CE < 59671 (E917h)	C - L, E - $+L
Yankee - 26	2756 (AC4h)	CEO	e	32 (20h) < C < 62199 (F2F7h), E - любой длины	C - L, E - $+L
Yankee - 27	2772 (AD4h)	CEO	e	32 (20h) < C < 62183 (F2E7h), E - любой длины	C - L, E - $+L
Yankee - 29	2932 (B74h)	CEO	e	32 (20h) < C < 62023 (F247h), E - любой длины	C - L, E - $+L
Yankee - 2A	2993 (BB1h)	CEO *	e	32 (20h) < C < (F207h), E - любой длины	C - L+4, E - $+L
Yankee - 2C	2881 (B41h)	CEO *	e	32 (20h) < C < 62071 (F277h), E - любой длины	C - L+4, E - $+L
Yankee - 2D	2879 (B51h)	CEO *	e	32 (20h) < C < 62055 (F267h), E - любой длины	C - L+4, E - $+L
Yankee - 2E	2977 (BA1h)	CEO *	e	32 (20h) < C < (F217h), E - любой длины	C - L+4, E - $+L
Yankee'-1905	1905 (771h)	CEO *	e	32 (20h) < C < 62071 (F277h), E - любой длины	C - L+4, E - $+L
Yankee'-1049	4096 (1000h)	CE	e*	32 (20h) < C < 63719 (F8E7h), E - не проверяет	C - L, E - $+L
YanShort-1624	1624 (658h)	E&.E	e	не проверяет	$+L
YanShort-1961	1961 (7A9h)	E&.E	e	не проверяет	$+L
512 (вер a-f)	512 (200h)	C	s*	512 (200h) < C < 65024(FE00h)	0

3 Зак. 2260

Таблица 1. Свойства и характеристики файловых и файлово-загрузочных вирусов. (продолжение).

Имя	Длина (L)	Заражаемые файлы	Место зараж	Диапазон длин заражаемых файлов	Увеличение длин файлов '$' – доп. до параграфа
Alabama	2884 (568h)	.E *	e*	не проверяет	1560
ATTENTION	393 (189h)	.C	e*	786 (312h) < C < 64921 (FD99h)	L + 1
Bebe	1004 (3ECh)	.C	e*	не проверяет	L
Datacrime	1168 (490h)	.C	e	не проверяет	L
Joker-01	29233 (7231h)	CEO *	e*	C < 6795 (1A8Bh), E < 6656 (1A00h)	C – L, E – 132+L
Kemerovo	256 (100h) *	.C	e*	C < 64767 (FCFFh)	L+1
Kuku-448	448 (1C0h) *	.C	s	453 (1C5h) < C < (F000h)	L ++
Lehigh	555 (22Bh)	COMMAND.COM	i	не проверяет	0
Liberty	2857 (B29h)	CE (boot)	e*	(500h) <C< (EFFFh), E< 640K (A0000)	L+$
LoveChild	488 (1E8h)	.C	e*	не проверяет	L
Loz-1023	1023 (3FFh)	.C&C	e	C < (FBFFh)	L
MGTU-273	273 (111h)	.C	e	не проверяет	L
Minsk-1075	1075 (433h)	.C.E	e	C < (FBBDh), 1024 (400h) < E	L
Mirror	482 (1E2h)	.C	e	не проверяет	L
MLTI-830	830 (33Eh)	.C	e	C < (FCBCh)	L
Peterburg	529 (211h)	C&.C	s	1744 (6D0h) <C< 61184 (EF00h)	L
RC-492	492 (1ECh)	C&.C	e*	не проверяет	L
Small-144	144 (90h)	CO	e	не проверяет	L
Sylvia	1301 (515h)	.C	s	C < (7530h)	L
VCOMM-637	637 (27Dh)	.E&E	e*	не проверяет	L + (0x200 *)
V-516	516 (204h)	C *	e*	не проверяет	L
V-707	707 (2C3h)	.C	e*	C < 61440 (F000h)	L
V-905	905 (389h)	E	e*	не проверяет	L
Wish-1024	1024 (400h)	E	e	512 (200h) < C < (EA60h), E – не проверяет	L
4096	4096 (1000h)	CEO	e*	C < 61440 (F000h)	L

По деструктивным возможностям вирусы можно разделить на:

а) безвредные, никак не влияющие на работу компьютера (кроме уменьшения свободной памяти на диске в результате своего распространения);

б) неопасные, влияние которых ограничивается уменьшением свободной памяти на диске и графическими, звуковыми и прочими эффектами;

в) опасные вирусы, которые могут привести к серьезным сбоям в работе компьютера;

г) очень опасные, которые могут привести к потере программ, уничтожить данные, стереть необходимую для работы компьютера информацию, записанную в системных областях памяти, и даже способствовать ускоренному износу движущихся частей механизмов (например, головок винчестера).

СПОСОБЫ РАСПРОСТРАНЕНИЯ И СТРУКТУРА КОМПЬЮТЕРНЫХ ВИРУСОВ

Ниже приводятся примеры структур и способов распространения файловых и загрузочных вирусов.

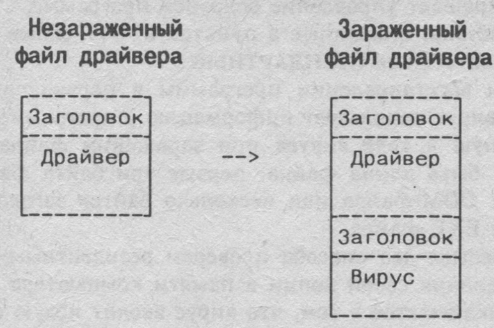
Файловые вирусы

Вирус может внедриться в файлы трех типов — командные файлы (BAT), загружаемые драйверы (SYS, в том числе IO.SYS и MSDOS.SYS) и выполняемые двоичные файлы (EXE и COM). Возможно внедрение вируса в файлы данных, но эти случаи возникают либо в результате ошибки вируса, либо при проявлении вирусом своих агрессивных свойств. Конечно, возможно существование вирусов, заражающих файлы, содержащие исходные тексты программ, библиотечные или объектные модули, но подобные способы распространения вируса слишком экзотичны и в дальнейшем не рассматриваются.

Мне известны два BAT-вируса. Они не представляют интереса, так как достаточно примитивны, очень просто обнаруживаются и удаляются.

Вирусы, внедряющиеся в SYS-файл, приписывают свои коды к телу файла и модифицируют его заголовок так, что DOS рассматривает инфицированный файл как цепочку из двух (или более) драйверов. Такой вирус может быть чрезвычайно опасным и живучим, так как внедряется в оперативную память при загрузке операционной

системы раньше любой антивирусной программы, если она, конечно, тоже не является драйвером.



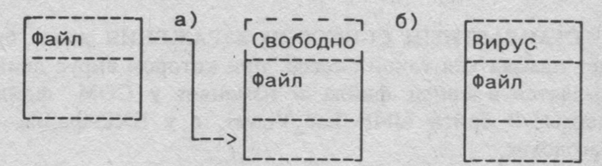
Аналогично вирус может записать свои коды в начало драйвера, а если в файле содержится несколько драйверов, то и в середину файла.

Выполняемые двоичные файлы, имеющие форматы COM или EXE, отличающиеся заголовком и способом запуска программ на выполнение. Расширение имени файла (".COM" или ".EXE") не всегда соответствует действительному формату файла, что, правда, никак не влияет на работу программы. Файлы COM и EXE заражаются по-разному, следовательно, вирус должен отличать файлы одного формата от файлов другого. Вирусы решают эту задачу двумя способами: одни анализируют расширение имени файла (".COM" или ".EXE"), другие — заголовок файла. Первый способ далее будет называться заражением .COM- (или .EXE-) файлов, второй — заражением COM- (или EXE-) файлов (обратите внимание на точку перед расширением!). В большинстве случаев вирус инфицирует файл корректно, то есть по информации, содержащейся в теле вируса, можно полностью восстановить зараженный файл.

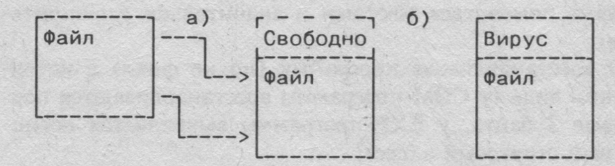
Файловые вирусы при распространении внедряются в тело заражаемого файла: в его начало, конец или середину.

Существует несколько способов внедрения вируса в середину файла: он может быть скопирован в таблицу настройки адресов, в область стека, может "раздвинуть" файл или переписать часть файла в его конец, а свои коды в освободившееся место, и так далее. Кроме того, копирование вируса в середину файла возможно в результате ошибки вируса — в этом случае файл может быть необратимо испорчен.

Известны два способа внедрения вируса в начало файла. Первый способ заключается в том, что вирус переписывает начало заражаемого файла в его конец, а сам копируется в освободившееся место. При заражении файла вторым способом вирус создает в оперативной памяти свою копию, дописывает к ней заражаемый файл и сохраняет полученную конкатенацию на диск.



Внедрение вируса в начало файла первым способом



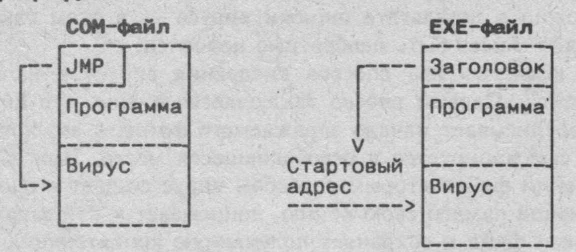
Внедрение вируса в начало файла вторым способом

Наиболее распространенным способом внедрения вируса в файл является дописывание вируса в конец этого файла. При этом вирус изменяет начало файла таким образом, что первыми исполняемыми командами программы, содержащейся в файле, являются команды вируса. В COM-файле в большинстве случаев это достигается изменением его первых трех (или более) байтов на коды инструкции JMP Loc_Virus (передача управления на тело вируса); EXE-файл либо переводится в формат COM-файла и затем заражается как COM-файл, либо выполняется модификация заголовка файла. В заголовке EXE-файла изменяются значение стартового адреса (CS:IP) и значение длины выполняемого модуля (файла), реже — регистры-указатели на стек (SS:SP), контрольная сумма файла и

Таблица 2. Свойства и характеристики загрузочных вирусов.

Имя	Длина	Заражаемые объекты	Уменьшение памяти (Кб)	Число сбойных секторов	Вызываемые эффекты
Brain/Ashar	3072 (C00h)	F	7	6	Ch
Brain/Singapore	3072 (C00h)	F	7	6	Ch
Ping-Pong	1024 (400h)	FB	2	2	V
Hacked Ping-Pong	1024 (400h)	FB	2	2	H
Ping-Pong modifyed	1024 (400h)	FB	2	2	V
Stone	512 (200h)	FM	2	0	VEr
Stone Rostov	512 (200h)	FM	2	0	HEr
Den Zuk	5120 (1400h)	F	7	0	VErCh
Disk Killer	3072 (C00h)	FB	8	Hard-0,flop-6	HV
Joshi	5120 (1400h)	FM	6	0	V
Liberty	4608 (1200h)	F (file)	10	0	H*
PrintScreen	512 (200h)	FB	2	0	Er*

др. Дополнительно к этому длины файлов перед заражением могут увеличиваться до значения, кратного параграфу (16 байт).



Внедрение вируса в конец файла

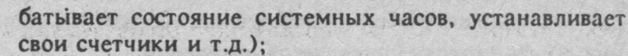
СТАНДАРТНЫМ СПОСОБОМ ЗАРАЖЕНИЯ далее будет называться такой способ, при котором вирус дописывается в конец файла и изменяет у COM- файла первые 3 байта (JMP Loc_Virus), а у EXE-файла — заголовок.

Вирус, после передачи ему управления, совершает следующие действия (приведен список наиболее общих действий вируса при его выполнении; для конкретного вируса список может быть дополнен, пункты могут поменяться местами и значительно расшириться):

а) восстанавливает программу (но не файл) в исходном виде (у COM-программы восстанавливаются первые 3 байта, у EXE-программы вычисляется истинный стартовый адрес);

б) если вирус резидентный, то он проверяет оперативную память на наличие своей копии и инфицирует память компьютера, если копия вируса не найдена. Если вирус нерезидентный, то он ищет незараженные файлы в текущем и (или) корневом каталоге, в каталогах, отмеченных командой PATH, и т.д., а затем заражает обнаруженные файлы;

в) выполняет, если они есть, дополнительные функции: какие-либо деструктивные действия, графические или звуковые эффекты и так далее. (Дополнительные функции резидентного вируса могут вызываться спустя некоторое время после активизации в зависимости от текущего времени, конфигурации системы, внутренних счетчиков вируса или других условий. В этом случае вирус при активизации обра-


батывает состояние системных часов, устанавливает свои счетчики и т.д.);

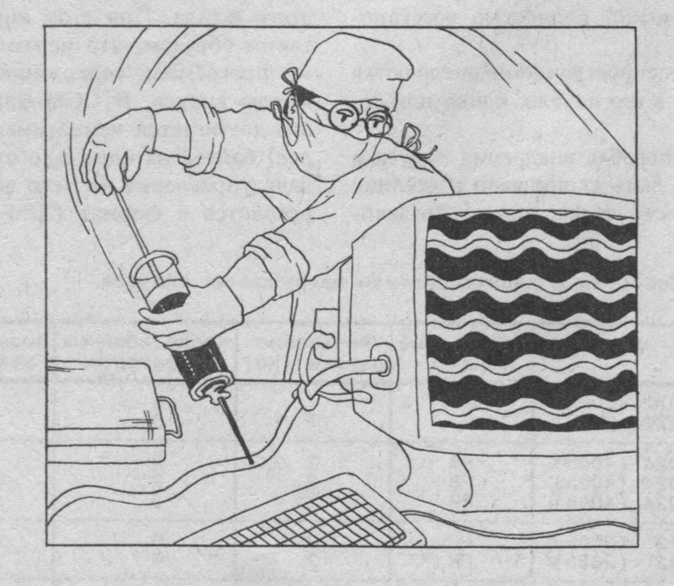
г) возвращает управление основной программе.

Действия, описанные в пунктах а)—г), далее будут упоминаться как СТАНДАРТНЫЕ.

При восстановлении программы в первоначальном виде, вирус использует информацию о программе, сохраненную в теле вируса при заражении файла. Это может быть длина файла, первые три байта файла в случае COM-файла или несколько байтов заголовка в случае EXE-файла.

Известны два способа проверки резидентным вирусом наличия своей копии в памяти компьютера. Первый заключается в том, что вирус вводит новую функцию некоторого прерывания, действие которой заключается в возврате значения "я здесь". При старте вирус обращается к ней, и, если возвращенное значение совпадает со значением "я здесь", то делается вывод, что память компьютера уже заражена и повторного заражения не производится. При проверке вторым способом вирус просто сканирует память компьютера. Оба способа могут в той или иной мере сочетаться друг с другом. При инфицировании оперативной памяти вирус ищет свободное место в памяти и записывает туда свою копию. Затем вирус переопределяет одно или несколько прерываний, необходимых ему для поиска заражаемых файлов, для выполнения деструктивных действий или звуковых и видеоэффектов.

При инфицировании файла вирус может производить ряд действий, маскирующих и ускоряющих его распространение. К подобным действиям можно отнести обработку атрибута read-only, снятие его перед заражением и восстановление после. Многие файловые вирусы считывают дату последней модификации файла и восстанавливают ее после заражения. Для маскировки своего распространения некоторые вирусы перехватывают прерывание DOS, возникающее при обращении к защищенному от записи диску (int 24h), и самостоятельно обрабатывают его.



DOS предусматривает единственный способ создания резидентных (TSR) модулей (помимо драйверов, указываемых в CONFIG.SYS) — при помощи функции KEEP (int 21h, ah = 31 или int 27h). Многие файловые вирусы для маскировки своего распространения используют другой способ: обрабаты-

вая системные области, управляющие распределением памяти (MCB), выделяют для себя свободный участок памяти, помечают его как занятый и переписывают туда свою копию. Некоторые вирусы внедряют свои TSR-копии в свободные участки памяти в таблице векторов прерываний, в рабочие области DOS, в память, отведенную под системные буфера.

Загрузочные вирусы

Загрузочные вирусы заражают загрузочный (Boot) сектор флоппи-диска и Boot-сектор или Master Boot Record (MBR) винчестера. При инфицировании диска вирус переносит оригинальный Boot-сектор (или MBR) в какой-либо другой сектор диска (например, в первый свободный) и копирует себя в загрузочный сектор (или в MBR). Если длина вируса больше длины сектора, то в заражаемый сектор помещается первая часть вируса, остальные его части размещаются в других секторах (например, в первых свободных). Если продолжение вируса размещается в первых свободных секторах диска, то они, как правило, помечаются как сбойные (точнее, сектора образуют кластер или кластеры, которые и помечаются как сбойные, так называемые псевдосбойные кластеры).

СТАНДАРТНЫМ СПОСОБОМ РАЗМЕЩЕНИЯ загрузочного вируса на диске далее будет называться такой способ, при котором продолжение вируса размещается в последовательных секторах диска, образующих псевдосбойные кластеры.



Известные мне загрузочные вирусы всегда резидентны. Они внедряются в память компьютера при загрузке с инфицированного диска. При этом системный загрузчик считывает содержимое первого сектора диска, с которого производится загрузка, помещает считанную информацию в память и передает на нее (т.е. на вирус) управление. После этого начинают выполняться инструкции вируса, который :

а) уменьшает объем свободной памяти (слово по адресу 0040:0013);
б) считывает с диска свое продолжение (если оно существует);
в) переносит себя в другую область памяти (например, в самые старшие адреса памяти);
г) устанавливает необходимые вектора прерываний;
д) совершает, если они есть, дополнительные действия;
е) копирует в память оригинальный Boot-сектор и передает на него управление.

Действия а)—е) далее будут упоминаться как СТАНДАРТНЫЕ.

В дальнейшем загрузочный вирус ведет себя так же, как и резидентный файловый вирус: перехватывает обращения операционной системы к дискам и инфицирует их, в зависимости от некоторых условий совершает деструктивные действия или вызывает звуковые или видеоэффекты.

ОПИСАНИЕ ВИРУСОВ

Ниже приводятся результаты анализа алгоритма работы имеющихся у меня файловых и загрузочных вирусов. Наиболее общие свойства вирусов представлены в таблицах каталога, дополнительная информация о каждом вирусе содержится в его описании.

Если свойство вируса отмечено в таблице, то в описании вируса оно может не упоминаться. Если файловый или загрузочный вирус совершают стандартные действия (см. выше), то описание этих действий может быть опущено.

Каталог вирусов

В предлагаемом каталоге для каждого вируса приведена информация о свойствах, описанных выше. Символьное представление этих свойств во многом повторяет классификационный код Н.Н.Безрукова.

В качестве основных опознавателей вируса будет указываться его имя и длина. Если вирус имеет несколько имен, то указываться будет одно из них, остальные известные мне имена приведены в табл. 4 каталога.

Длиной файлового вируса считается длина тела вируса (коды+данные+стек вируса, если он есть) без всяких добавок. В большинстве случаев она равна минимальному приращению длин файлов при их заражении. Для загрузочных вирусов в качестве длины принимается полная длина тела вируса, т.е. число занимаемых вирусом секторов, умноженное на число байт в секторе.

В таблицах каталога используются такие символы:
+ — наличие свойства;
- — отсутствие свойства;
* — в описании имеется дополнительная информация;
& — наличие различных свойств;
V — вызывает видеоэффекты (Video);

M — вызывает звуковые эффекты (Music);
H — имеет деструктивную функцию (Harm);
Er — содержит ошибку, которая может проявляться как деструктивная функция (Error);
Ch — изменяет служебную информацию: на диске — имя диска, в DOS — возвращает неверную версию DOS и т.п. (Change).

Для файловых вирусов (табл. 1):
C — COM-файлы;
.C — .COM-файлы (файлы с расширением ".COM");
E — EXE-файлы;
.E — .EXE-файлы (файлы с расширением ".EXE");
S — SYS-файлы;
O — прочие файлы (Other).

Для загрузочных вирусов (табл. 2):
B — Boot-сектор винчестера;
M — MBR винчестера;
F — Boot-сектор флоппи-дисков.

Для файлово-загрузочных вирусов употребляется полный набор символики.

Например, если для файлового вируса в графе, соответствующей поражаемым объектам, указано "E C&.C", то это означает, что вирус поражает только файлы, имеющие либо формат EXE, либо формат COM и одновременно расширение имени ".COM" (т.е. не поражаются файлы, которые имеют формат COM и расширение имени ".EXE").

При описании свойств файловых вирусов (табл. 1) используются специальные знаки.

Место внедрения в файл:
s — начало (start);
e — конец (end);
i — середина (insert).

Метод заражения:
FIND — используя функции FindFirst и FindNext;
INT — перехватывая обращения DOS к файлам;
INFI — используя оба способа.

Способ внедрения в оперативную память (только для резидентных вирусов):
MCB — оперируя с MCB;
KEEP — используя прерывания DOS (int 21h f.31 или int 27h);
IntTable — записывается в таблицу векторов;
Buffer — записывается в дисковый буфер.

Длина:
L — длина вируса;
$ — длина заражаемого файла увеличивается до значения, кратного параграфу;
++ — возможно повторное заражение файлов.

Файловые вирусы

Семейство "Amstrad"

Семейство (в моей коллекции — 6 штук) нерезидентных опасных вирусов. Общие черты:
— заражаются все .COM-файлы в текущем оглавлении;
— вирус записывает себя в начало файла;
— возможно повторное заражение;
— файлы большой длины могут быть уничтожены.

Некоторые версии вируса быстро проявляются: начиная с пятого "поколения" вируса, при старте зараженной программы с вероятностью 1/2 на экране появляется сообщение: "Programm sick error: Call doctor or buy PIXEL for cure description". Одна из версий выдает на экран политический лозунг на болгарском языке.

Семейство "Anti-Pascal"

Семейство (5 штук) очень опасных нерезидентных вирусов. Заражают или портят не более 2-х файлов во всех каталогах на текущем диске и диске C:. При поиске незараженных файлов используется рекурсивный обход дерева каталогов. Заражают .COM-, .BAK- и .PAS файлы. В случае COM-файла вирус записывается в его конец (версии вируса длин 400, 440 и 480) или начало (остальные версии вируса). При заражении .BAK и .PAS-файлов записывается в начало файла, при этом старое начало не сохраняется, то есть файл оказывается безвозвратно потерянным. Некоторые версии вируса переименовывают .BAK и .PAS файлы в .EXE.

Семейство "April 1st" ("SURIV")

"April 1st COM"

Файловый резидентный опасный вирус. Поражает .COM-файлы (кроме COMMAND.COM) при их запуске на выполнение (int 21h, ax = 4B00h). Опасен — не проверяет длин файлов. Записывается в начало файла. При заражении:
а) создает файл TMP$$TMP.COM;
б) записывает в него свою копию;
в) дописывает в него заражаемый файл;
г) уничтожает заражаемый файл;
д) присваивает файлу TMP$$TMP.COM имя заражаемого файла.

При создании своей резидентной копии использует функцию 31h прерывания 21h (по схеме "Иерусалимского" вируса).

Проявляется начиная с 1 апреля 1988 года при заражении файлов: выводит на экран текст "APRIL 1ST HA HA HA YOU HAVE A VIRUS" и завешивает систему; в последующие дни сообщает "YOU HAVE A VIRUS !!". Перехватывает int 21h. Помимо перечисленных, содержит строки "sURIV" (в начале зараженного файла), "COMMAND.COM", "TMP$$TMP.COM".

"April 1st EXE"

Файловый резидентный опасный вирус. Поражает .EXE-файлы при их запуске на выполнение (int 21h, ax = 4B00h). Опасен — не вполне корректно работает с длиной файла. При заражении внедряется в середину файла между заголовком и загружаемым модулем, при этом вирус:
а) создает файл TMP$$TMP.EXE;

б) считывает из заражаемого файла первые 1Bh байтов заголовка, модифицирует в них байты, соответствующие длине модуля, стартовым значениям CS, IP, SS, SP, контрольной сумме файла (устанавливается значение 1984h); затем записывает модифицированный заголовок в файл TMP$$TMP.EXE;

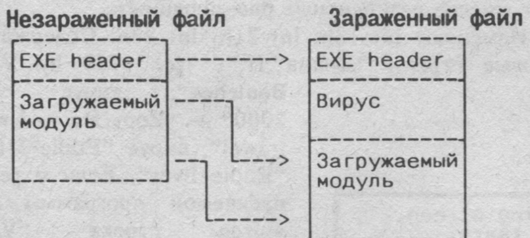
в) копирует из заражаемого файла в TMP$$TMP.EXE таблицу настройки адресов (ТНА), модифицируя ее описанным ниже способом;

г) дописывает в файл TMP$$TMP.EXE копию вируса;

д) дописывает в файл TMP$$TMP.EXE загружаемый модуль инфицируемого файла;

е) уничтожает заражаемый файл;

ж) присваивает TMP$$TMP.EXE имя заражаемого файла.



Так как загружаемый модуль файла при заражении смещается на расстояние, равное длине вируса, то вирусу приходится производить соответствующие изменения в ТНА: у каждого элемента ТНА содержимое байтов, соответствующих смещению сегмента, увеличивается на значение, равное длине вируса в параграфах.

При создании своей резидентной копии использует функцию 31h прерывания 21h (по схеме "Иерусалимского" вируса).

Начиная с 1 апреля 1988 года при активизации расшифровывает (XOR FFh) и выводит текст "APRIL 1ST HA HA HA YOU HAVE A VIRUS" и завешивает систему; в последующие дни текст не появляется, а система зависает приблизительно через *55* минут после активизации. Перехватывает int 21h и, в зависимости от текущего дня, может перехватывать int 1Ch. Содержит строки "sURIV" и "TMP$$TMP.EXE".

Семейство "Eddie"

"Eddie-651"

Инфицирует COM- и EXE-файлы при загрузке их в память для выполнения (int 21h, ax=4B00h). У заражаемого файла устанавливает значение секунд в 1Fh. Никак не проявляется. Содержит строку "Eddie lives". Изменяет вектор прерывания 21h.

Вирус перехватывает функции DOS FindFirst и FindNext FCB (int 21h, ah=11h, 12h) и "уменьшает" длины зараженных файлов, т.е. вирус невозможно определить по изменению длин файлов (если, конечно, он резидентно находится в памяти). Утилиты, которые не используют указанные функции DOS (например, Norton Commander), а напрямую используют содержимое секторов, хранящих каталог, показывают правильную длину зараженных файлов. При "уменьшении" длин файлов вирус определяет их зараженность по значению секунд даты последней модификации (62 секунды) и не проверяет истинных длин файлов. Если такое значение секунд встретится у файла небольшой длины (меньше 651 байт), то команда DIR выдаст несколько странную длину файла — около 4 гигабайт.

"Eddie-1800"

Очень опасный резидентный файловый вирус. Быстро размножается: инфицирует COM- и EXE-файлы при загрузке их в память для выполнения, при создании, переименовании, открытии и закрытии файла (int 21h, ax=4B00h или ah=3Ch, 5Bh, 56h, 3Dh, 2Eh).

При создании своей копии в оперативной памяти компьютера вирус пытается занять область памяти с самыми старшими адресами, разрушая временную часть командного интерпретатора COMMAND.COM. По окончании работы программы командный интерпретатор COMMAND.COM восстанавливает свою временную часть, при этом происходит открытие файла COMMAND.COM и, соответственно, его заражение. Таким образом, наиболее подверженным заражению файлом является COMMAND.COM.

Вирус предпринимает ряд эффективных мер для того, чтобы остаться незамеченным.

1) При старте любой программы вирус помечает программный сегмент как последний и становится невидимым для этой программы, по окончании работы программы вирус помечает программный сегмент как не последний.

2) Аналогично вирус поступает с вектором прерывания 21h при старте программы: он восстанавливает его первоначальное значение.

3) Вирус не позволяет программам изменять вектор прерывания 21h и защищается таким образом от резидентных антивирусов, установленных после того, как активизировался вирус.

4) Вирус пытается обойти программы, следящие за прерыванием 13h, и установить вектор этого прерывания в его первоначальное значение.

Вирус очень опасен: периодически стирает сектор со случайным номером ("оплевывает" диск), при этом он может уничтожить часть информации на диске. Вирус анализирует 2 байта (8-й и 10-й) Boot-сектора диска, с которого была запущена зараженная программа. Вирус увеличивает на 1 значение 10-го байта и сохраняет его в Boot-секторе. Если новое значение 10-го байта кратно 16, то вирус стирает на диске сектор со случайно выбранным номером, зависящим от значения 8-го байта.

Изменяет вектора прерываний 13h, 21h, 27h. Содержит текстовые строки "Eddie lives...somewhere in time", "Diana P.", "This program was written in the city of Sofia (C) 1988-89 Dark Avenger".

"Eddie-2000" и "Eddie-2100"

Очень опасные резидентные файловые вирусы.

Быстро размножаются: инфицируют файлы при загрузке их в память для выполнения, при создании, переименовании и закрытии файла, при чтении или изменении его атрибутов (int 21h, ax=4B00h или ah=3Ch, 3Eh, 43h, 56h, 5Bh).

Вирусы следят за тем, чтобы длины заражаемых файлов увеличивались ровно на 2000 (или 2100) байт (к EXE-файлам, помимо выравнивания на границу параграфа, дописывается необходимое число байт). У заражаемых файлов (как и при поражении вирусом "Vienna") устанавливается новое значение секунд даты последней модификации — 62 секунды.

Вирус "Eddie-2000" предпринимает ряд эффективных мер для того, чтобы остаться незамеченным, многие из них повторяют аналогичные меры вируса "Eddie-1800". Маскирующую функцию выполняет длина вируса: достаточно трудно заметить приращение длины, кратное 1000 байтам. Введена и другая функция защиты: вирус перехватывает функции DOS FindFirst и FindNext FCB (int 21h,ah=11h,12h) и "уменьшает" длины зараженных файлов на 2000 байт.

Вирус "Eddie-2100" дополнительно к этому обрабатывает функции FindFirst и FindNext ASCII (ah=4Eh,4Fh); исправлена ошибка, из-за которой команда DIR сообщала, что длина файла около 4 гигабайт (см. описание вируса "Eddie-651"). Вирус непонятным мне образом манипулирует с секторами винчестера — видимо, пытается либо активизировать, либо вылечить неизвестный мне загрузочный вирус.

Вирусы "Eddie-2000" и "Eddie-2100" очень опасны: аналогично вирусу "Eddie-1800" стирают сектора со случайными номерами. При этом вирусы обращаются напрямую к драйверу, обслуживающему диск, и обходят многие резидентные блокировщики.

Изменяют int 13h, int 21h, int 27h. Содержат текстовые строки "Диана П.", "(c) 1989 by Vesselin Bontchev"; вирус "Eddie-2000" — "Zopy me — I want to trawel", вирус "Eddie-2100" — "Eddie lives". Если в теле запускаемой программы содержится строка "Vesselin Bontchev" (Веселин Бончев — автор известных болгарских антивирусов), то вирус зацикливается и система зависает.

Таблица 3. Названия вирусов.

Cascade	Falling Letters, Letterfall, Crying screen, Rash, Dropper, Листопад, Falling tears, Herbist, Autumn, 1704/Cascade, Blackjack
VACSINA-NN Yankee-NN.	TP-nn, где nn - десятичное представление числа NN
VACSINA-NN	Vacsina, Vacs, Vacsina sound, Sina
Yankee-NN	Doodle, Yankee Doodle
Yankee-2C	Five O'clock Music, Ibris, Music Player, Neww, Jumbo Brawler
Vienna-648	Timebomb, Rebooter, Reset, DOS-62, Timestamp, Restart, VHP-648, Frier, System reboot, UNESCO
Vienna-534	Toothless, Microsoft 1988, Microsofttyright Micro88, Vienna
Vienna-757	Кузьмич
Amstrad	Cancer, IV, Ams
Eddie-1800	Dark Avenger, Sofia, Pyramides, Diana, Dav Eddie
Eddie-2000	V2000, Bch, Bontchev, Диана П.
Jerusalem	Black Hole, Black Friday, Israeli, Hebrew, Time, Jeru
SunDay	9-Sunday
YanShort-1961	Father-1961, Doodle2
4096	4K, 100 Years, Frodo, Hiding, Century
April 1st	Suriv, April
Ping-Pong	Ball, Italian Ping-Pong ver.B, Bouncing Ball Bouncing dot, Ping, Vera Gruz
Brain/Ashar	Pakistani Brain/Ashar
Disk Killer	Killer, Ogre
Stone	New Zealand, Mariguana
Den Zuk	Search, Venezuelan

Семейство "Cascade"

Мне известны два представителя семейства: "Cascade-1701" и "Cascade-1704". Их тела, за исключением начала (32 байта), зашифрованы. В качестве ключа используется длина зараженного файла, поэтому два "штамма" одного вируса в большинстве случаев будут совпадать только на первых 32 байтах.

При выполнении зараженной программы управление с помощью команды JMP передается на начало вируса. Первыми командами вирус определяет длину исходного файла и расшифровывает свое тело.

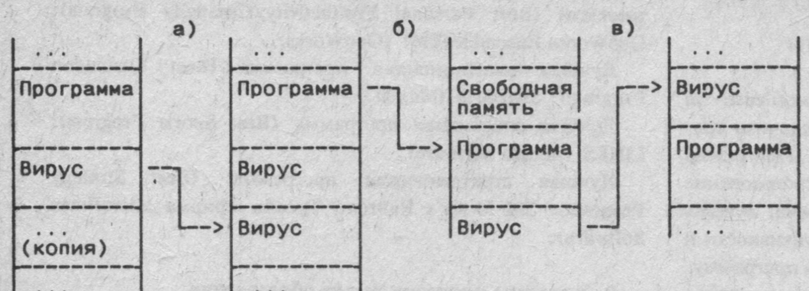
При создании своей резидентной копии вирус:

а) копирует свое тело в старшие адреса памяти;

б) перемещает в старшие адреса тело основной программы;

в) перемещает свое тело в освободившееся над телом основной программы пространство;

г) устанавливает int 1Ch, int 21h и int 28h на свою копию.



Вирус заражает только COM-файлы при загрузке их в память для выполнения (int 21h, ax=4B00h). Инфицирование происходит стандартным способом. Наиболее распространенные версии этих вирусов не производят повторного заражения файлов.

Изменяют вектора прерываний 1Ch, 21h и 28h. Вызывают характерный видеоэффект: осыпание букв на экране. Не имеют деструктивных функций.

Заранее благодарю всех, кто пришлет свои замечания, предложения или сообщит об обнаруженных в тексте неточностях.

"Доктор" Е. Касперский
Москва, 8 января 1991 г.

4 Зак. 2260

Ассоциация Software Publishers Association присудила свои ежегодные дипломы

Диплом за выдающуюся программу (Excellence in Software Award) получила графическая многозадачная среда GeoWorks Ensemble под Windows и OS/2. Программа победила в четырех категориях: лучшее использование компьютера, лучшая пользовательская программа, лучшая программа для повышения персональной продуктивности и диплом критиков за лучшую пользовательскую программу.

Microsoft Windows 3.0 получили 2 диплома — лучшая новая программа для бизнеса и диплом критиков в этой же категории.

SimEarth (фирма Maxis) и Ashlar Vellum (фирма Ashlar) также получили по два диплома.

Члены Ассоциации выбирали из 115 продуктов, представленных 63 компаниями, тогда как дипломы критиков присуждались по результатам голосования представителей прессы без ограничений по номенклатуре программ.

Стив Возняк, сооснователь фирмы Apple, получил диплом за все свои достижения в компьютерной области (lifetime achievement award).

Ниже следует полный список врученных дипломов.

1. Дипломы делового программного обеспечения:

Лучший новый программный продукт для бизнеса (Best New Business Software Product): Microsoft Windows 3.0 (фирма Microsoft).

Лучшая прикладная программа для бизнеса (графической или дисплейной ориентации) (Best Business Application: Graphic or Display Orientation): allCLEAR (фирма CLEAR Software).

Лучшая прикладная программа для бизнеса (цифровой или информационной ориентации) (Best Business Application: Numeric or Data Orientation): Quattro Pro 2 (Borland International).

Лучшая прикладная программа для бизнеса (текстовой ориентации) (Best Business Application: Word or Text Orientation): Ami Pro (фирма Lotus Development).

Лучшее средство для программирования, (Best Programming Tool): Turbo C++ Professional (Borland International).

Лучшая утилита (Best Utility Program): The Norton Utilities 5.0 (фирма Symantec).

Лучшая программа для вертикального рынка (Best Vertical Market Application): Ashlar Vellum (фирма Ashlar).

2. Дипломы для пользовательского программного обеспечения:

Лучшая. потребительская программа (Best Consumer Program): GeoWorks Ensemble(TM) (GeoWorks).

Лучшая активная программа (Best Action/Arcade Program): Faces...Tris III (Spectrum HoloByte).

Лучшая программа-приключенческая игра (Best Fantasy Role Playing/Adventure Program): King's Quest V (фирма Sierra On-line).

Лучшее средство для повышения персональной продуктивности (Best Personal Productivity/Creativity Program): GeoWorks Ensemble(TM) (GeoWorks).

Лучшая имитационная программа (Best Simulation Program): SimEarth (Maxis).

Лучшая спортивная программа (Best Sports Program): LINKS (Access Software).

Лучшая стратегическая программа: (Best Strategy Program): Sid Meier's Railroad Tycoon (фирма MicroProse Software).

3. Дипломы программам для образования:

Лучшая программа для обучения (Best Education Program): Compton's MultiMedia Encyclopedia (stand-alone version) (фирма Britannica Software).

Лучшая программа для начального обучения (Best Early Education Program): Mixed Up Mother Goose (Multimedia) (Sierra On-Line).

Лучшая программа для начальной школы (Best Elementary Education Program): NumberMaze Decimals & Fractions (Great Wave Software) и Super Solvers Outnumbered! (The Learning Company).

Лучшая программа для домашнего обучения (Best Home Learning Program): Deluxe Edition of Where in the World is Carmen Sandiego (фирма Broderbund Software).

Лучшая программа для повышения продуктивности школьного обучения (Best School Productivity/Creativity Program): The Children's Writing and Publishing Center (IBM networked version) (The Learning Company).

Лучшая программа для средней школы (Best Secondary Education Program): SimEarth (Maxis).

Лучшая программа для удовлетворения специальных запросов (Best Special Needs Program): The Sentence Master: Level 1 (фирма Laureate Learning Systems).

4. Дипломы всем программам:

Лучшее новое использование компьютера (Best New Use of A Computer): GeoWorks Ensemble(TM) (GeoWorks).

Лучший пользовательский интерфейс у новой программы (Best User Interface in A New Program): Ashlar Vellum (фирма Ashlar).

5. Дипломы критиков:

Лучшая пользовательская программа (Best Consumer Program): GeoWorks Ensemble (GeoWorks).

Лучшая деловая программа (Best Business Program): Microsoft Windows 3.0 (фирма Microsoft).

Лучшая учебная программа (Best Education Program): Compton's Multimedia Encyclopedia (Britannica Software).

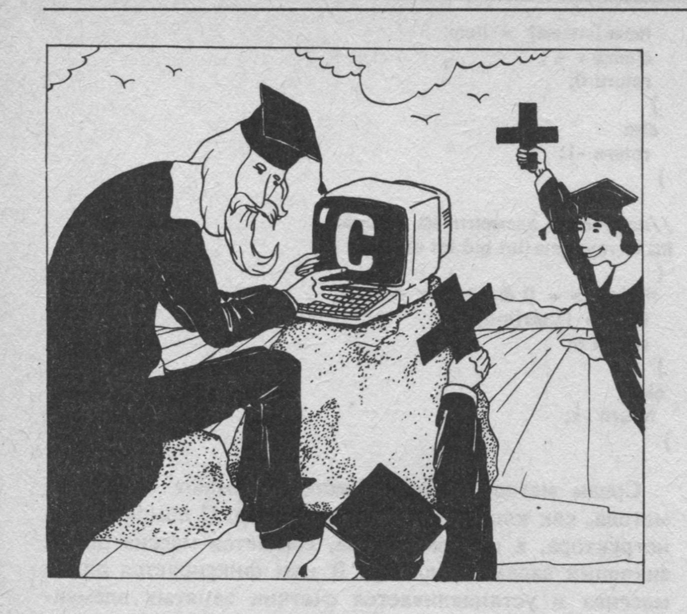
6. И наконец, дипломы компьютерным журналистам:

Лучшему аналитику: Keith Ferrell, OMNI Magazine.

Лучшие выпуски новостей: Michael Rogers, Newsweek.

Лучшие обзоры программ: Johnny Wilson, Computer Gaming World.

Newsbytes News Network, March 15, 1991.
Newsbytes News Network, March 22, 1991.



Язык С++ появился как расширение языка Си, предназначенное для поддержки объектно-ориентированного программирования. Интерес к нему постоянно растет, но для того, чтобы в полной мере использовать его возможности, необходимы специальные знания в этой области. В противном случае программирование на нем сводится к модификации программирования на языке Си. Ниже дается краткий обзор специфических языковых средств С++ и иллюстрируется их применение.

Язык С++ и объектно-ориентированное программирование

Объектно-ориентированное программирование

Объектно-ориентированное программирование появилось в результате развития методов структурного программирования и методов программирования с использованием абстракции данных.

В структурном программировании для разбиения системы на составляющие элементы применяется функциональная декомпозиция. Этот процесс обычно осуществляется "сверху-вниз", в результате чего детализируется реализация функций системы. Само разбиение строится и оценивается на основе использования мер модульности—связности и прочности модулей. Такой метод слабо учитывает влияние обрабатываемых структур данных на архитектуру программы.

В подходе, основанном на использовании абстракции данных, главное внимание уделяется выбору представления данных, наиболее подходящего для реализации решаемой задачи. На основе выбора осуществляется модульная декомпозиция программ. Широко используется понятие типа и строится иерархия типов на базе детализации представления данных на различных уровнях декомпозиции. При таком методе существует разрыв между структурами данных и функциями для работы с ними. При использовании объектно-ориентированного программирования этот разрыв уничтожается. Центральным элементом в данном случае является объект, представляющий собой модель некоторого элемента реального мира и содержащий в себе как данные, так и операции над ними.

Для построения объектов используются абстрактные типы данных и скрытие информации. Большое внимание уделяется взаимодействию между объектами.

Объекты одного типа объединяются в классы, которые представляют собой основные "строительные блоки" разрабатываемых программ. В классах интегрируются как определения данных, так и определения функций, выполняемых над данными.

Для успешного применения объектно-ориентированного метода к разработке программ необходимо наличие подходящего языка программирования. В настоящее время такими языками являются: SmallTalk, Turbo Pascal v.6.0, C++ и др.

Дадим краткий обзор элементов языка С++, обеспечивающих объектно-ориентированное программирование.

Классы и их структура

Класс является основным элементом языка С++, обеспечивающим объектно-ориентированное програм-

мирование. Он представляет собой расширение структур языка Си. В классах помимо определений данных допускаются определения выполняемых над ними функций. Кроме того, в них имеются средства управления доступом к данным, позволяющие организовывать взаимодействие различных классов друг с другом и классов с обычными функциями. Функции, входящие в класс, часто называются методами.

Определение класса иллюстрирует следующий пример:

```
//     iarray.h
class array
{
 int maxitems;   //максимальное число элементов
 int citems;     //число элементов, размещенных в массиве
 int *items;     //указатель на массив
public:
 //конструктор
 array(int nitems);
 //деконструктор
 ~array();
 //занесение элемента в массив
 int putitem(int item);
 //получение элемента из массива
 int getitem(int ind,int & item);
 //получение фактического числа элементов в массиве
 int count({return citems;});
};
```

В определении даны объявления внутренних переменных класса (переменные состояния), а также прототипы методов, обеспечивающих работу с данными.

Приведенный класс предназначен для создания целых массивов заданной длины и для размещения в них целых значений. Назначение переменных пояснено соответствующими комментариями.

Класс iarray содержит в себе пять методов. Прототипы четырех из них даны в файле irray.h, а для пятого метода дана полная реализация.

Сами исходные тексты методов класса iarray приведены ниже:

```
// iarray.cop
#include "iarray.h"

//конструктор
iarray::iarray(int nitems)
 {
 items = new int[nitems];
 maxitems = nitems;
 citems = 0;
}

//деструктор
iarray::~iarray()
 {
  delete items;
 }

//занесение элемента в массив
int iarray::puta(int item)
 {
  if (citems < maxitems){
   items[citems] = item;
   citems++;
   return 0;
  }
  else
   return -1;
 }

//получение элемента из массива
int iarray::geta(int ind,int & item)
 {
 if (ind >= 0 && ind < citems){
   item = items[ind];
   return 0;
 }
 else
  return -1;
}
```

Среди методов особое место занимают такие два метода, как конструктор и деструктор. С помощью конструктора, в данном случае, создается массив целых значений заданной длины. В нем фиксируется длина массива и устанавливается счетчик занятых элементов, а также производится распределение памяти под массив. Последняя операция выполняется введенным в язык С++ оператором new. Он проще, чем функция malloc, так как при его применении не требуется задавать размер выделяемой памяти в байтах и преобразовывать полученный указатель к конкретному типу.

Второй специальный метод — деструктор. С его помощью уничтожается созданный конструктором объект. Для уничтожения динамически созданных объектов в языке С++ используется оператор delete, выполняющий роль функции tree языка Си.

Методы putitem и getitem обеспечивают контролируемое обращение к элементам массива. В первом методе отслеживается возможность размещения нового элемента в массиве. Во втором — правильность задания индекса получаемого из массива элемента.

Метод count позволяет получить число включенных в массив элементов. Он реализован прямо в исходном тексте заголовочного файла iarray.h.

Отметим, что при определении методов за пределами текста класса используется операция "::", например:

```
int iarray::putitem ()
```

Эта операция говорит о том, что областью действия метода putitem является класс uarray.

Коснемся теперь использования определенного нами класса. Можно привести следующий пример

```
#include "array.h"
#include <iostream.h>
#include <stdlib.h>
#include <time.h>

int main()
 {
 array    m(10);
 int      i,j,k:
 int      n;
```

```
randomize();
for (i = 1;i <= 5;i++){
 m.putitem(rand());
}
n - m.count();
for (i = 0;i <= n;i++){
 m.getitem(i,j);
 cout << j <<"\n";
}
}
```

В этом примере создается *объект m*, принадлежащий классу iarray. Далее в массив m вводится пять целых чисел, полученных путем обращения к датчику случайных чисел. Эти числа вводятся в выходной поток cout, которому соответствует дисплей.

В данном примере мы сталкиваемся с важнейшим понятием *объект*. Объектом является переменная m, объявленная как принадлежащая классу iarray. Объект — это конкретный экземпляр массива целых чисел.

Действия с объектами осуществляются с помощью методов, обращение к которым задается составными именами, например,

m.putitem();

или

m.getitem();

Отметим, что в объявлении

iarray m(10);

также происходит вызов конструктора, в котором собственно и выполняются действия по созданию массива.

Есть другая возможность создания объекта с помощью оператора new. Например, в следующем фрагменте программы

```
iarray   *a;
iarray   *b;
int      n1 = 10;
int      n2 = 20;

a = new iarray(n1);
b = new iarray(n2);
. . . .
delete a;
delete b;
```

динамически создаются два массива a и b, размерностью 10 и 20 соответственно.

Связь класса с внешней средой

В объектно-ориентированном программировании широко используются абстрактные типы данных и скрытие информации. Следовательно, проблема взаимодействия классов со средой, в которой они используются, приобретает первостепенное значение.

Решается эта проблема путем управления доступом к элементам классов с помощью использования в определениях классов ключевых слов private, public и protected.

Описатель private говорит о том, что следующие за ним составляющие элементы класса скрыты внутри него и недоступны непосредственно во внешней среде. Обращение к ним допускается только внутри класса с помощью его методов.

Описатель public говорит о том, что следующие за ним элементы класса доступны во внешней среде. Обычно public указывается перед методами класса.

Отметим, что элементы класса не могут иметь описателей типа памяти automatic, extern. Они могут иметь дополнительно только описатель static. Элемент класса с описателем static будет общим для всех объектов данного класса.

Помимо public и private в классах можно использовать описатель friend, с помощью которого объявляются "дружественные" ему функции и целые классы. Если в классе объявлен прототип функции с описателем friend, не принадлежащий ему, то эта функция получает право доступа к внутренним элементам класса, например

```
class type {
 private:
  int prm;

 public:
  friend void ftype (type x, int y);
};

void ftype (type x, int y);
 {
  x.prm = y;
 }
```

В данном примере в "дружественной" классу a функции ftype осуществляется доступ к внутренней переменной prm объекта x, принадлежащего к классу type.

Отношения между классами

Одной из важнейших черт объектно-ориентированного программирования является наличие механизма наследования свойств одного класса другими классами. Он позволяет строить новые классы на базе ранее созданных и этим самым способствует повторной используемости результатов процесса программирования.

Программа может включать в себя набор базовых классов, которые не связаны ни с какими другими. На основе базовых классов строятся производные классы, которые наследуют от базовых классов их структуры данных и методы. Таким образом, производные классы становятся расширением базовых, при этом они не включают в себя детали реализации базовых классов.

Рассмотрим пример, иллюстрирующий механизм взаимодействия между базовым и производными классами:

```
//      basearr.h

class basearray
 {
 protected:
  int        maxitems; // максимальное число элементов
```

```
 int         citems; //число размещенных элементов

public:
 //конструктор
 basearray(int nitems) {maxitems = nitems; cinems = 0;};
};

//      iarray.h

class iarray:public basearray
{
 private:
  int        *items;
 public:
  //конструктор
  iarray(int nitems);
  //деструктор
  ~iarray ();
  //занесение элемента в массив
  int putitem(int item);
  //получение элемента из массива
  int getitem(int ind,int & item);
  //получение числа элементов, размещенных в массиве
  int count ({return citems;});
 };
```

Исходный текст методов класса iarray приведен ниже

```
//       iarray.cpp

 #include "basearr.h"
 #include "iarray.h"

//конструктор
iarray::iarray(int nitems): basearray(nitems)
 {
  items = new int[nitems];
 }

//деструктор
iarray::~iarray()
 {
  delete items;
 }

//занесение элемента в массив
int iarray::putitem(int item)
 {
  if (citems < maxitems){
   items[citems] = item;
   citems++;
   return 0;
  }
  else
   return -1;
 }

//получение элемента из массива
int iarray::getitem(int ind,int & item)
 {
  if (ind >= 0 && ind < citems){
   item = items[ind];
   return 0;
  }
  else
   return -1;
 }

//получение числа элементов, размещенных в массиве
int iarray::count(){return citems;};
```

В новой версии реализации массива выделен базовый класс basearray, в котором определены общие для массивов составляющие, независимо от типов элементов. В производном классе определен конкретный тип массива. В нашем случае тип массива int, хотя можно определить тип массива как float, как struct и т.п.

Класс iarray наследует от класса basearray конструктор. При создании объекта типа iarray в конструкторе iarray вызывается конструктор basearray.

С помощью механизма наследования осуществляется скрытие информации. Следует обратить внимание на то, что в базовом классе появился новый описатель управления доступом — protected. Его наличие говорит о том, что следующие за ним переменные будут доступны в производном и недоступны в классах и функциях, не принадлежащим к этим классам.

Улучшение изобразительных свойств языка

В языке С++ реализован ряд новшеств, позволяющих во многих случаях улучшить читаемость программ и упростить их написание.

Прежде всего отметим расширение способов передачи параметров в функции. Возможна передача параметров не только по значению, но и по имени. Рассмотрим такой пример

```
void sum (int *mas, int n, int &s);
int main ()
{
 int s;
 int mas[5] = {1,2,3,4,5};

 sum (mus, 5, s);
 printf ("%d\n", s);
}

void sum (int *mas, int n, int &s);
{
 int i;

 s = 0;
 for (i=0; i,n; i++)
  s = s + mas[i];
 return;
}
```

Параметр s в функции sum и в объявлении ее прототипа объявлен как передаваемый по имени. Это избавляет от необходимости работы с указателями.

Другим ценным свойством языка С++ является возможность замещения операции и функций. Следую-

Имеется класс записей:

```
class rec
{
 private:
  int a;
  int b;
  char    str[10];
 public:
  rec (int x1, int x2)
  {
   a = x1;
   b = x2;
   *str = 0;
  }
  void add (rec & p1, rec p2)
  {
   p1.a = p1.a + p2.a;
   p1.b = p1.b + p2.b;
  }
};
```

Если в программе объявлены два объекта

```
rec    r1(1,2);
rec    p2(3,4);
```

то можно выполнить суммирование r1 и r2 следующим образом:

```
add (r1, r2);
```

Можно определить метод сложения двух объектов типа rec следующим образом:

```
rec operator +  (rec & p1, rec p2)
{
 p1.a = p1.a + p2.a;
 p1.b = p1.b + p2.b;
  return;
 }
```

Теперь объекты класса rec складываются следующим образом:

```
p1 = p1 + p2;
```

Выше кратко рассмотрены некоторые возможности языка C++, используемые в объектно-ориентированном программировании. Их применение в полной мере требует изучения как конструкций языка, так и методов проектирования и реализации объектно-ориентированных программ.

Этой проблеме посвящена книга "Объектно-ориентированная разработка программ на языке C++", издание которой планируется в текущем году. В ней будут детально показаны возможности системы Turbo C++ для реализации программ, связанных с обработкой экономической информации. В качестве примеров рассматриваются классы, обеспечивающие работу с клавиатурой, дисплеем ПЭВМ типа IBM PC, классы для работы в оперативной памяти (строки переменной длины, списки, очереди), классы для выполнения диалогового ввода-вывода, классы для управления программами, классы для обеспечения связи с файловой системой MS DOS и др.

Пример взаимодействия классов строится на реализации простого редактора текстов.

Заказы на книгу Черноусова Е.А. "Объектно-ориентированная разработка программ на языке C++" просим направлять по адресу: 103062, Москва, а/я 86

Е.Черноусов

Введение в MS Windows

Характерные примеры работы

Итак, мы разобрали почти все фундаментальные понятия MS Windows. Пройдено больше половины пути. Чтобы подвести некоторый итог, давайте рассмотрим вполне конкретный пример на языке иллюстраций. Этот пример должен закрепить ваши знания визуально.

Рассмотрим сначала конкретный пример работы с файлом. Работа с файлом осуществляется посредством Меню Файла (File), входящего практически в любую программу для MS Windows.



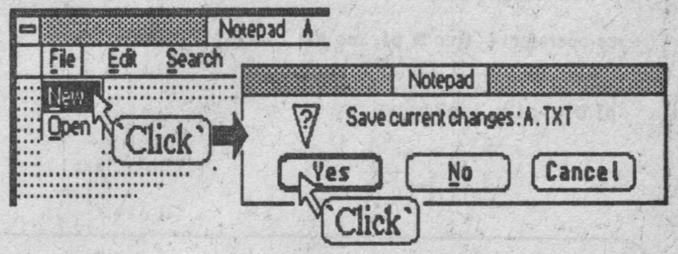
Сначала рассмотрим подробно меню File (Файл):

Команда	Действие
New (новый)	Удаляются все существующие данные, удаляется имя файла из прямоугольника заголовка. Программа стартует с самого начала. Если команда может испортить существующие данные, выдается соответствующая панель сообщения.
Open... (открыть)	С помощью модальной панели диалога пользователь указывает имя файла и затем открывает этот файл в качестве текущего.
Save (запомнить)	Запоминает существующие данные в текущем файле. Если файл не указан, т.е. данные вводились после команды New, то автоматически генерируется команда Save As...
Save As... (запомнить в)	С помощью модальной панели диалога пользователь указывает имя файла, куда будут записываться данные. Далее производится запись файла на диск.

Главы из книги А.Н.Никольского и В.Ю.Назарова "Введение в MS Windows", выпускаемой издательством "Финансы и статистика" в 1991 г. Продолжение. Начало в № 4.

Теперь промоделируем некоторые часто встречающиеся ситуации при работе с меню Файл.

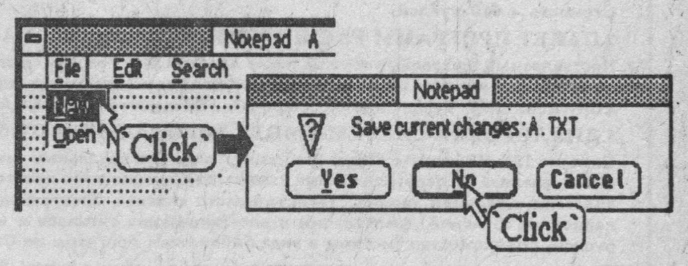
1. Выбирается команда New, но на экране имеются вводимые ранее данные, связанные с файлом A.TXT. Данные надо запомнить.



"Запомнить ли изменения в файле A.TXT?"
Ответ: "Да"

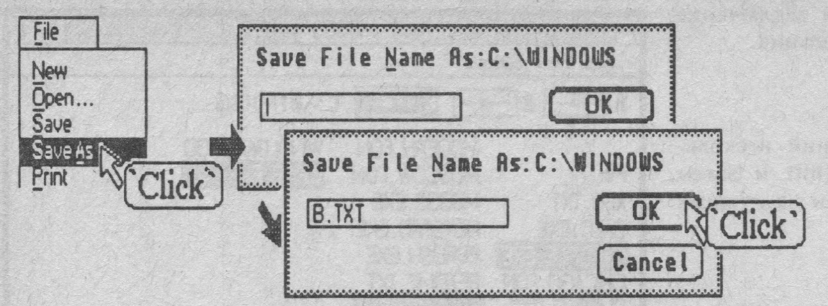
Alt, N → Enter

2. Теперь рассмотрим предыдущий пример, но данные запоминать не будем.

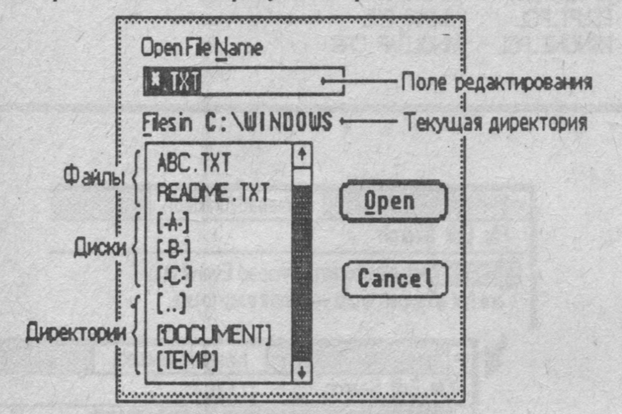


"Запомнить ли изменения в файле A.TXT?"
Ответ: "Нет"

3. Рассмотрим пример, когда после выбора команды New введены данные и их требуется запомнить в файле B.TXT.



4. Определенного навыка требует работа с полями списка. Активизируем, например, команду Open ... и выберем из списка требуемый файл.



Опишем поля панели диалога, активизируемой командой Open... Поле редактирования текста предназначено для ввода имени файла или типа файла. Поле списка содержит все файлы указанного типа, входящие в текущую директорию; имена дисков в квадратных скобках и имен поддиректорий текущей директорий в квадратных скобках. Если произвести двойной щелчок над именем диска, то поле списка будет переформировано, исходя из списка файлов указанного типа и директорий, принадлежащих корневой директории этого диска. Двойной щелчок можно заменить обычным щелчком и активизацией командной кнопки Open. Аналогично будет изменен список, если произвести двойной щелчок над именем поддиректории. В список будут включены все файлы указанного типа и поддиректории той директории, над которой был произведен щелчок.

Выбор файла осуществляется двойным щелчком над именем этого файла в поле списка. Просмотр всего списка производится с помощью линий прокрутки поля списка или клавиш **PgUp**, **PgDn**, ↑ , ↓, если фокус установлен на поле списка. При выборе файла двойной щелчок в клавиатурном варианте заменяется на установку маркера выбора на требуемый файл, а затем нажатие клавиши Enter.

5. В качестве последнего примера рассмотрим пример, когда требуется запомнить введенные данные в файле B.TXT, который уже существует на диске.

Тем, кто раньше никогда не сталкивался с MS Windows, лучше всего, конечно, познакомиться со средой в реальной работе. Тогда все приводимые в книге примеры и описания станут более очевидными и понятными. Данная книга не является теоретическим описанием или философским трактатом — она описывает предмет реальной действительности, который можно пощупать руками, увидеть, как он работает, почувствовать. И это, непременно, следует сделать.

Выделение

Понятие выделения в MS Windows тесно связано с понятием клипборда (Clipboard). Клипборд — это специальный буфер для временного хранения информации. Копируемая в клипборд информация должна быть сначала выделена, а затем уже выполняется команда Копировать(Copy). Выделяться может как текстовая, так и графическая (рисунки, графики) информация.

Выделение может также производиться с целью изменения каких-либо свойств выделяемой информации. Характерным примером здесь может являться замена в тексте шрифтов (фонтов).

Выделение используется для указания нескольких параметров при выполнении команды такой, например, как "Копировать Файл". Данная команда может за один сеанс производить копирование как одного файла, так и группы выделенных файлов.

Выделение производится с помощью мыши или клавиатуры. Для пользователя процесс визуального выделения заключается в маркировании требуемой информации. При выделении текстовых данных маркирование производится путем реверсивной закраски требуемой области. При выделении графической инфор-

5 Зак. 2260

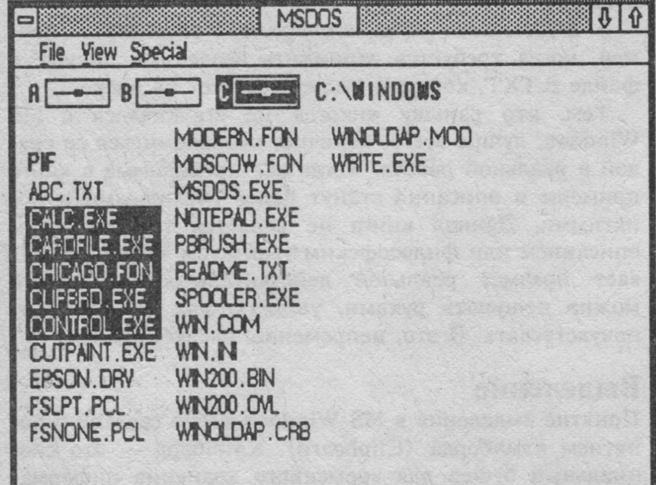
мации маркирование представляет собой заключение требуемых данных в габаритный прямоугольник.

Одновременное выделение названий нескольких файлов

Для одновременного выделения названий нескольких файлов следует нажать клавишу **Shift** и затем осуществить щелчок левой клавиши мыши на нужных названиях в списке файлов.

Одновременное выделение названий нескольких файлов

Для выделения блока (непрерывной последовательности), файлов следует нажать клавишу **Shift** и с помощью функциональных стрелок ↑, ↓, ←, →, осуществить выделение блока. Процесс выделения оканчивается после отпускания клавиши **Shift**.



Выделение "рассыпанных" файлов осуществляется следующим образом:

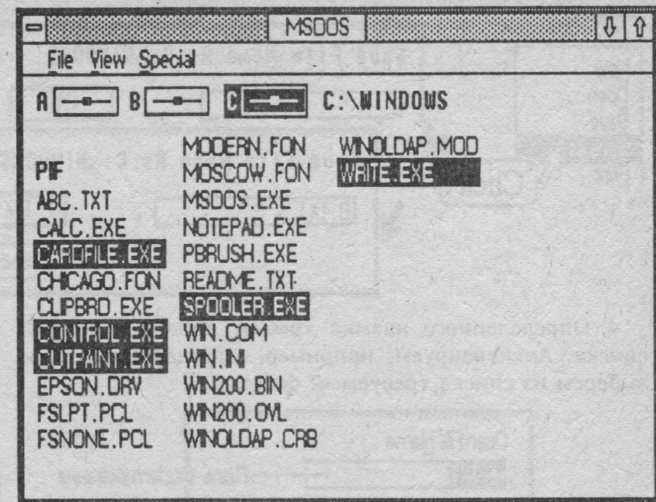
1. Нажать клавишу **Ctrl**.
2. Не отпуская **Ctrl**, с помощью функциональных стрелок подвести маркер к требуемому файлу.
3. Отпустить **Ctrl**.
4. Нажать **Space** для выделения файла.
5. Повторить шаги 1-4 для выделения следующих файлов.

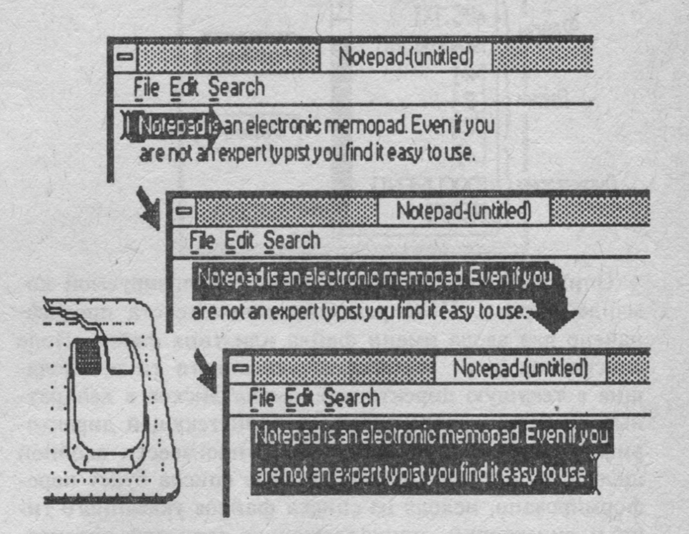
Ниже приводится иллюстрация, в которой наглядно отображено выделение нескольких "рассыпанных" файлов.

Выделение текста

Для выделения текста следует нажать левую клавишу мыши в месте начала выделения и затем, не отпуская клавишу, осуществлять перемещение мыши. При этом в соответствии с перемещением будут изменяться границы реверсивной области закраски. Процесс выделения завершается освобождением левой клавиши мыши.

Ниже приведена иллюстрация, в которой рассмотрены три последовательных фрагмента выделения текста.





Выделение текста

Подвести курсор мыши к месту начала выделения текста. Нажать клавишу **Shift** и, не отпуская ее, использовать функциональные стрелки. При этом будут изменяться границы реверсивной области закраски. Процесс выделения завершается высвобождением клавиши **Shift**.

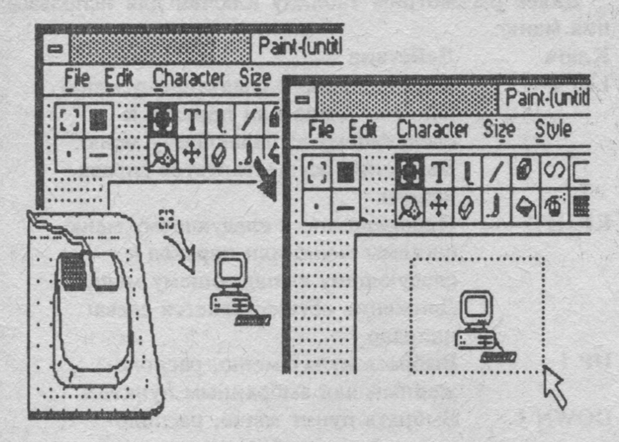
Выделение текста производят, когда следует изменить шрифтовое оформление, произвести копирование фрагмента текста или его удаление, а также в ряде других случаев, когда выделенный текст должен представлять из себя единый неделимый информационный блок.

Выделение текста чаще всего производится в текстовых процессорах и настольных издательских системах. В MS Windows недостатка в обработчиках текста не имеется — среди них Word for Windows, Aldus PageMaker, Ami Professional, Write.

Выделение графической информации

Процесс выделения происходит по тем же принципам, что и процесс выделения текста.

Курсор подводится к месту, откуда будет начинаться выделение, нажимается левая клавиша мыши и производится перемещение мыши. При этом изменяется габаритный прямоугольник выделения. Процесс заканчивается после того, как пользователь отпускает левую клавишу мыши.



На иллюстрации представлены два фрагмента экранов. Первый экран отображает начальный момент выделения. Второй экран показывает заключительный этап. Результатом выделения в итоге явилось изображение компьютера.

Многозадачность MS Windows. Обмен данными между задачами

Мы уже упоминали о том, что в MS Windows можно запускать одновременно несколько программ. Каждая программа будет высвечивать на экране свое основное окно и в результате мы увидим приблизительно следующую картину:

Эта иллюстрация представляет собой набор перекрывающихся окон. Среди этих окон существует только одно активное окно, сфокусированное на работу с клавиатурой и мышью. Это окно располагается над всеми остальными окнами.

Активизация окна

Самым простым способом активизации требуемого окна является щелчок левой клавиши мыши в области этого окна.

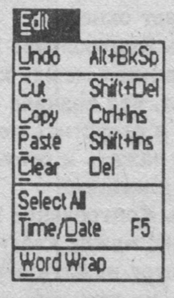
Активизация окна

Активизация окна с помощью клавиатуры производится комбинациями **Alt + Esc**, **Alt + Shift + Esc**, **Alt + Tab**, **Alt + Shift + Tab**. Данные комбинации клавиш являются переключателями активных окон.

Активизируемое окно перерисовывается над всеми окнами и его прямоугольник заголовка и окантовка приобретает окраску, характерную для активного окна.

В MS Windows можно одновременно запускать одну и ту же программу несколько раз. При этом в каждом из окон одной и той же программы могут обрабатываться различные файлы.

В условиях многозадачности приобретает важное значение обмен данными между задачами. Здесь мы сталкиваемся еще с одним фундаментальным понятием MS Windows — буфером обмена данными (Clipboard). Буфер обмена данными будем называть клипборд. Механизм передачи данных от программы к программе в MS Windows крайне прост. Передающая эти данные программа копирует эти данные в клипборд, а программа-получатель эти данные из клипборда забирает. Такая передача осуществляется через "Меню Редактирования" (Edit Menu).



Команда	Действие
Undo (отменить)	Отменяет действие последней произведенной команды.
Cut (вырезать)	Копирует выделенные данне в клипборд с одновременным их удалением из области данных
Copy (копировать)	Копирует выделенные данные в клипборд без удаления.
Paste (вставить)	Копирует содержимое клипборда в область данных программы.

Clear (очистить)	Удаляет выделенные данные из области данных, оставляя при этом клипборд без изменения.

Заключительные главы, описывающие основные принципы работы с MS Windows, будут посвящены работе с клавиатурой и мышью. В основном мы уже неплохо изучили этих устройств в среде MS Windows. Но теперь давайте остановимся на них более профессионально, дадим полный набор функциональных возможностей работы и с тем, и с другим устройством.

Клавиатурный интерфейс

Если у вас нет желания пользоваться мышью или просто-напросто нет таковой в наличии, MS Windows предоставляет вам развитый клавиатурный интерфейс, предназначенный для активизации программы, окна или панели диалога; для доступа к системе меню; для перемещения окна, изменения его размеров, а также для многих других функций.

Зарезервированные системные ключи

Зарезервированные системные ключи представляют собой комбинации клавиш и используются в MS Windows для реализации различных системных команд. Ключи не могут быть переназначены ни одной из программ и активны в течение всего сеанса работы MS Windows. В частности, некоторые акселераторы являются такими ключами. Рассмотрим сводную таблицу зарезервированных системных ключей.

Ключ	Действие
Alt+F4	Закрывает окно
Alt, Space	Выбрать Контрольное Меню активного окна (запятая между клавишами означает, что вначале следует нажать и отпустить первую клавишу, а затем нажать и отпустить вторую клавишу).
Shift+Esc	Выбрать Контрольное Меню активного окна ("+" между клавишами означает, что следует нажать первую клавишу, а затем, не отпуская ее, нажать вторую клавишу).
Alt+Tab	Переключение между текущим активным окном и окном, которое было активным до него. При переключении икона восстанавливается до окна.
Alt+Shift+Tab	То же самое, но в обратном направлении.
Alt+Esc	Переключиться на следующую запущенную программу, двигаясь в прямом направлении последовательности запуска программ.
Alt+Shift+Esc	Переключиться на предыдущую запущенную программу, двигаясь в обратном направлении последовательности запуска программ.
Alt+F5	Восстановить окно до его первоначального размера.
Alt+F7	Перемещение окна.
Alt+F8	Изменение размеров окна.
Alt+F9	Убрать окно и высветить вместо него икону.
F10 или Alt	Перейти к системе меню.

Далее рассмотрим таблицу ключей для использования меню:

Ключ	Действие
LEFT—	Перемещение к следующему меню системы меню или переход к следующему выпадающему меню. Движение осуществляется справа налево.
RIGHT—	Перемещение к следующему меню системы меню или переход к следующему выпадающему меню. Движение осуществляется слева направо.
UP I	Выбрать пункт меню, расположенный над выбранным пунктом.
DOWN I	Выбрать пункт меню, расположенный под выбранным пунктом.
Enter	Выполнить команду, соответствующую выбранному пункту меню.
Esc	Отказаться от работы с меню.

Следующая таблица описывает ключи для работы с панелями диалога:

Ключ	Действие
Tab	Перемещение к следующему полю или группе. Движение осуществляется слева направо и сверху вниз.
Shift+Tab	Перемещение к следующему полю или группе. Движение осуществляется справа налево и снизу вверх.
Space	Активизация командной кнопки. Изменение состояния выключателя.
Enter	Выйти из панели диалога, активизировав выделенную командную кнопку.
Esc	Отменить работу панели диалога.

Следующая таблица описывает ключи для выполнения функций редактирования, перемещения и выделения:

Ключ	Действие
Tab	Перемещение к следующему пункту слева направо и сверху вниз. Перемещение текста вправо на один табулятор.
Shift+Tab	Перемещение к следующему пункту справа налево и снизу вверх. Перемещение текста влево на один табулятор.
Enter	Выполнить команду, установленную по умолчанию. Начать новый абзац в документе.

←, →, ↑, ↓	Перемещение в соответствии со значением функциональной стрелки.
Ctrl+←	Переместиться влево или в начало группы или секции.
Ctrl+→	Переместиться вправо или в конец группы или секции.
Ctrl+↑	Переместиться вверх или в верхнее положение группы или секции.
Ctrl+↓	Переместиться вниз или в нижнее положение группы или секции.
Shift+Ctrl+←	Переместиться и расширить выделение влево или в начало поля, группы или секции.
Shift+Ctrl+→	Переместиться и расширить выделение вправо или в конец поля, группы или секции.
Shift+Ctrl+↑	Переместиться и расширить выделение вверх или в начало поля, группы или секции.
Shift+Ctrl+↓	Переместиться и расширить выделение вниз или в конец поля, группы или секции.
PgUp	Осуществить прокрутку вверх на один экран.
PgDn	Осуществить прокрутку вниз на один экран.
Ctrl+PgUp	Осуществить прокрутку влево на один экран.
Ctrl+PgDn	Осуществить прокрутку вправо на один экран.
Shift+PgUp	Осуществить прокрутку и расширить выделение вверх на один экран.
Shift+PgDn	Осуществить прокрутку и расширить выделение вниз на один экран.
Shift+←	Переместиться и расширить выделение влево на одну единицу измерения.
Shift+→	Переместиться и расширить выделение вправо на одну единицу измерения.
Ins	Переключатель моды вставка/ замена.
Shift+Ins	Вставить содержимое клипборда.
Ctrl+Ins	Копировать выделенную информацию в клипборд.
Del	Удалить выделенную информацию.
Ctrl+Del	Удалить с текущей позиции до конца строки.
Shift+Del	Удалить выделенную информацию и поместить ее в клипборд.
Backspace	Удалить символ или пробел слева от курсора.
Alt+ Backspace	Отменяет последнее действие редактирования.
Home	Перемещение в начало строки.
Ctrl+Home	Перемещение в начало данных.
Shift+Home	Переместиться и расширить выделение до начала линии.
Shift+ Ctrl+Home	Переместиться и расширить выделение до начала данных.
End	Переместиться в конец строки.
Ctrl+End	Переместиться и расширить выделение до конца строки.
Shift+ Ctrl+End	Переместиться и расширить выделение до конца данных.

Следует уяснить одно простое правило: клавиша Shift используется для выделения необходимой информации.

Интерфейс мыши

В MS Windows использование мыши является альтернативным методом управления. Это не является ограничением ее применения, а наоборот, практически всегда использование мыши делает управление более простым и доступным.

Управление мышью фактически сводится к четырем действиям:

Действие	Описание
Установка (point)	Передвинуть мышь, чтобы курсор мыши указывал на требуемое место или поле.
Щелчок (click)	Быстро нажать и отпустить клавишу мыши.
Тащить (drag)	Нажать левую клавишу мыши и, не отпуская ее, осуществлять перемещение мыши.
Двойной щелчок (double-click)	Быстро щелкнуть клавишей мыши дважды.

В MS Windows мышь используется: для доступа к меню; для активизации программы; для перемещения окна, изменения его размеров; для закрытия окна или трансформации окна в икону и т.д.

Обычно во всех программах используется только левая клавиша мыши. Другие клавиши применяются в ситуациях, оговариваемых в специальных инструкциях к программам.

Рассмотрим подробно использование мыши в ситуациях выделения информации и выбора пунктов.

При работе по выделению текста существуют следующие соглашния об использовании мыши:

Действие	Описание
Click Щелчок	Перемещает текстовый курсор в место, на которое указывает курсор мыши.
Двойной щелчок	Выделить слово.
ВЫДЕЛЕНИЕ Тащить	Выделять текст, начиная с того места, где клавиша мыши была нажата, до места, где клавиша отпускается.
Shift+Click	Осуществляет выделение от текстового курсора до того места, куда указывает курсор мыши.
Ctrl+ ВЫДЕЛЕНИЕ	Первоначально выделяется область в границах от текстового курсора до курсора мыши, а затем при перемещении мыши данная область выделения изменяется.

Виды курсоров мыши

В зависимости от типа выполняемого действия форма курсора мыши может изменяться, тем самым сигнализируя о выполнении соответствующего типа действия.

Приведем самые распространенные виды курсоров мыши:

Вид	Описание
↖	Курсор мыши в виде стрелки. Имеет место при работе с меню, панелями диалога, линиями прокрутки и т.д.
I	Соответствует текстовому курсору.
⌛	Курсор — песочные часы. Сигнализирует о процессе обработки информации и задержке выдачи данных на экран.

Некоторые подробности об окнах

Как мы уже рассматривали, каждая программа имеет свое основное окно. Это и только это окно идентифицирует программу. Сколько запущено программ, столько и будет на экране окон. Некоторые окна могут быть невидимыми, так как либо другие окна их перекрывают, либо эти окна заменены иконами (спрятаны в иконы).

Помимо основного окна программа в случае необходимости может открывать сопутствующие ее работе дополнительные окна. Эти окна называются вторичными (child) окнами. Вторичные окна могут иметь те же атрибуты, что и основные окна, кроме меню. Количество открываемых вторичных окон лимитируется только техническими возможностями компьютера, но не идеологией MS Windows.

Понятие фокуса и активного окна полностью распространяется на вторичные окна. Управление вторичными окнами совпадает с управлением вторичными окнами. Панели диалога и сообщений в конечном итоге также являются вторичными окнами, но выделяются в отдельный класс, поскольку имееет ярко выраженные специфические характеристики в виде полей и кнопок.

При работе со вторичными окнами следует всегда помнить о том, что при закрытии основного окна закрываются все вторичные окна, принадлежащие этому окну.

Самая последняя глава

Итак, мы рассмотрели все основные понятия MS Windows и немного научились работать с этой средой. Если вам показалось приведенное описание слишком сложным и отпугнуло от MS Windows, не отчаивайтесь. Наверняка вы оказались в такой ситуации, когда у вас не было под рукой самой среды MS Windows или вы все время читали эту книгу в переполненном автобусе или метро. Вам просто сложно было переключиться от будничных дел к освоению MS Windows. Не торопитесь делать поспешные выводы — отложите ненадолго эту книгу, а затем, через некоторое время снова к ней вернитесь. Среда MS Windows действительно достойна вашего внимания! Хорошие знания об этой среде никогда не помешают вам в будущем.

Следующая часть этой книги посвящена описанию конкретных программ для MS Windows. Среди них текстовый процессор, электронная картотека и ряд других программ.

Следующая часть уже может рассматриваться вами как справочное руководство. Опыт работы с MS Windows показывает, что знание основополагающих принципов работы среды позволяет в самые короткие сроки, практически без описания освоить любую программу, работающую в MS Windows. Чем и хороша настоящая среда, что стандартизует весь пользовательский интерфейс независимо от специфики решаемых задач.

(Продолжение следует).

Зарплата работающих в американской компьютерной индустрии растет...

По данным исследований, проведенных международной ассоциацией менеджеров сервиса (AFSMI), сервис, в том числе и информационный, в последнее время приносит все возрастающую прибыль многим компаниям, что позволяет увеличивать заработную плату работающих в AFSMI.

Среднегодовая зарплата директора по техническому обслуживанию выросла с 35500 долл. в 1980 году до 70700 долл. в 1990 году.

Среди технических работников самой высокооплачиваемой должностью остается старший инженер по аппаратному обеспечению. Зарплата на этой должности за последние 10 лет выросла с 24 до 43 тысяч долл. в год. Заработная плата других представителей этой профессии в 1990 году была следующей: старший инженер по обслуживанию пользователей — 32500, старший техник-ремонтник — 28700, техник-ремонтник — 23000 долл. в год.

В разных местах зарплата также существенно различается. В Калифорнии она самая высокая.

Newsbytes News Network, March 15, 1991.

Несмотря на то, что объявленная в Сан-Хосе, Калифорния, выставка изделий для Windows и OS/2 состоялась, последним в экспозиции места не нашлось. Впрочем, посетителей это ничуть не огорчило.

Computerworld, March 11, 1991.



В наше время истинным бедствием для компьютеров и их программного обеспечения стали многочисленные компьютерные вирусы. По мере накопления опыта борьбы с вирусами начали возникать целые антивирусные технологии работы с компьютерами. Одним из эффективных средств предохранения от возможных потерь информации является защита логических дисков.

Программная защита дисков

Защита логических дисков винчестера

Едва ли не самой обсуждаемой темой в мире IBM-совместимых персональных компьютеров становится проблема компьютерных вирусов. Существует несколько типов программ, предназначенных для борьбы с вирусами. Самые необходимые и широко распространенные из них — это детекторы и доктора вирусов. Один из наиболее популярных в нашей стране пакетов подобного типа — Scan/Clean ассоциации McAfee. Имеются также полифаги, сочетающие в себе оба указанных свойства, лучшим примером которых будет, несомненно, AIDSTEST Д.Лозинского. Однако у всех таких программ есть одно уязвимое место — они неспособны бороться против неизвестных им вирусов, как неспособны и защитить другие программы от заражения ими. Следовательно, необходимо попробовать защитить свои файлы, хранящиеся на винчестере, против атак вирусов другим способом.

Идея создания резидентной антивирусной программы родилась почти одновременно с появлением самих компьютерных вирусов. Первые такие программы действительно эффективно обнаруживали попытки нелегальной записи на диск. Однако авторы вирусов достаточно быстро справились с таким "досадным недоразумением" и перестали пользоваться прерыванием 21h для записи в программу своего кода, обратившись напрямую к прерыванию 13h. Перестроились и антивирусные программы, которые стали проверять теперь именно это прерывание.

Но вот в последнее время появились сообщения о создании двух новых классов компьютерных вирусов. Первые из них используют функцию 1300h прерывания 2Fh для получения первоначального адреса прерывания 13h, имевшегося еще до загрузки DOS. Вторые вначале проходят процедуру записи в пошаговом режиме, добираясь до адресов BIOS, перехватывают управление и в дальнейшем осуществляют запись, обращаясь уже непосредственно к контроллеру.

Для успешной борьбы с современными вирусами антивирусная программа должна обладать высокими "интеллектуальными" способностями, что, как правило, сопряжено с большим объемом исполняемого кода. В результате такая программа может сильно замедлять работу с винчестером и было бы нежелательно использовать ее постоянно. Поэтому к серьезной антивирусной программе целесообразно обращаться лишь в случае возникновения реальной опасности заражения. В остальное же время очень полезной могла бы оказаться маленькая резидентная программа, способная защищать винчестер от "пиратских" попыток записи на уровне прерывания 13h. Эту задачу и выполняет предлагаемая вашему вниманию программа PROTECT.

PROTECT является резидентной программой защиты винчестера от записи. Защищается не весь винчес-

тер, а лишь отдельные его участки, определяемые цилиндрами, на которые распространяется запрет записи. Такой подход позволяет осуществлять вызов PROTECT из файла AUTOEXEC.BAT и защищать только системные логические диски винчестера. Даже в случае если на вашем диске уже имеются вирусы, они не смогут произвести свое разрушительное действие и заразить новые программы. При этом все остальные логические диски остаются открытыми для записи и на них можно спокойно работать. Программа PROTECT полезна также и против "троянских" программ, разрушающих системную информацию.

У программы PROTECT есть одно слабое место — она загружается через DOS и поэтому не может препятствовать заражению винчестера boot-вирусами. Тем не менее, вы можете избежать и этой беды, используя программу Vitamin-B (автор А. Сесса), которая разработана для вакцинации областей BOOT и Partition Table. Некоммерческая версия программы распространяется через Софтпанораму. Последняя версия избавлена от досадных ошибок, которые ранее в отдельных случаях приводили к уничтожению некоторых логических дисков, при работе с DOS версий старше 3.30. Программа Vitamin-B является отличным средством защиты от boot-вирусов; комбинируя ее вместе с PROTECT, вы получите на своем компьютере программную среду, устойчивую к вирусным атакам.

PROTECT не имеет ключей запуска. Это связано с требованиями минимального размера и максимального быстродействия резидентной программы, непрерывно проверяющей все операции с винчестером. Вместо ключей вы просто указываете количество защищаемых цилиндров в приведенном ниже исходном тексте программы и заново компилируете ее (например, Turbo Assembler´ом). Для минимизации кода программы в ней не предусмотрены какие бы то ни было встроенные средства отключения защиты. Эту задачу выполняет отдельная программа PROTOFF, которая сама не загружается резидентно, а лишь проверяет при своем запуске наличие PROTECT в памяти компьютера и переключает защиту из активного состояния в пассивное и наоборот.

В целях борьбы с вирусами нового поколения PROTECT осуществляет вызов функции 1300h прерывания 2Fh, заменяя в таблице старый адрес прерывания 13h ложной информацией. Если в дальнейшем какой-либо вирус попытается обойти PROTECT, то он получит неверный адрес прерывания 13h и неизбежно "завесит" компьютер, пытаясь провести запись на диск. На мой взгляд, нажатие кнопки RESET не будет чрезмерной платой за успешное отражение вирусной атаки. А связанные с этим потери информации будут неизмеримо меньшими в сравнении с теми разрушениями, которые вам причинил бы проникший в систему вирус.

Если у вас есть необходимость защитить логические диски, расположенные на винчестере не по порядку, а выборочно, то код программы легко модифицировать, изменив условия проверки на те или иные группы цилиндров.

Необходимо отметить, что программа PROTOFF настроена на смещение ключевого байта, вырабатываемое при компиляции Turbo Assembler´ом версии 2.0 и старше. Если вы пользуетесь другим компилятором, то возможно, вам придется исследовать Turbo Debugger´ом или каким-либо другим отладчиком типа Quaid Analyzer работу PROTECT и найти в ней то место, где размещается ключевой байт. При наличии отладочной информации по программе это не составит большого труда.

Следует также иметь в виду, что если поверх PROTECT вы разместите другую резидентную программу, которая переустанавливает вектор прерывания 13h на себя, то PROTOFF не сможет найти PROTECT в памяти и выдаст сообщение об ошибке. Будьте внимательны при размещении в памяти ваших резидентных программ — и у вас не возникнет никаких проблем с работой комплекса PROTECT.

Ю. Кравацкий

```
; Защита логических дисков винчестера методом
; установки альтернативной программы обработки
; прерывания 13h, запрещающей запись на указанные
; цилиндры.
; Program was written by JVK Software.
; All rigths reserved. Апрель 1991
;
; Эта программа является freeware и может
; передаваться в неизмененном виде всем
; пользователям в некоммерческих целях. Для
; контакта с автором звоните по телефонам:
; 135-99-07 (Москва), 477-13-24 (Киев)
; Кравацкий Юрий Всеволодович

sys_size equ     400           ; Количество защищаемых
                               ; цилиндров

code    segment para public
        assume  CS:code

        org     100h

start:  jmp     install

int13h  proc    near

        cli                    ; Запретить прерывания

; Проверка прерывания 13h на все функции записи
        cmp     AH,03h         ; write sector (запись сектора)
        je      nextchk
        jmp     run13h
        cmp     AH,0Fh         ; AT write sector (запись
                               ; сектора на AT)
```

```
        je      nextchk
        cmp     AH,0Bh  ; write long (длинная запись)
        jne     run13h
nextchk:
        cmp     DL,80h  ; Проверка обращения к hard
                        ; drive 0 (первому винчестеру)
        jne     run13h
        push    CX
        rol     CL,1
        rol     CL,1
        xchg    CL,CH
        and     CX,03FFh
        cmp     CX,sys_size ; Находимся ли мы в
                        ; защищаемых пределах
        jge     runhere ; Нет
        pop     CX
        mov     CS:save_al,AL
        pop     AX
        mov     CS:ret_off,AX
        pop     AX
        mov     CS:ret_seg,AX
        pop     AX      ; Сохранить флаги
        mov     AL,CS:save_al
        mov     AH,3    ; Выставить для DOS ошибку
                        ; "Write protect"
        sti             ; Разрешить прерывания
        stc             ; Установить Carry-флаг для
                        ; индикации ошибки

        db      0EAh    ; Первый байт инструкции
                        ; far jump
ret_off dw      0
ret_seg dw      0

runhere:
        pop     CX
run13h:
        sti

        db      0EAh    ; Первый байт инструкции
                        ; far jump
old13o  dw      0       ; В этих двух переменных
old13s  dw      0       ; хранится старый адрес
                        ; прерывания 13h
save_al db      0

int13h  endp

install proc    near

        push    DS      ; Этот блок предназначен для
        push    ES      ; борьбы с вирусами,
        mov     AX,1300h ; использующими ту же
        int     2Fh     ; функцию для получения
        cmp     DX,BX   ; адреса прерывания 13h,
                        ; существовавшего до
                        ; загрузки DOS. После этого
                        ; прерывание 2Fh будет
                        ; возвращать ложный адрес
                        ; прерывания 13h.
        jne     already_load
        mov     AX,2513h ; Восстановление
        int     21h     ; разрушенного адреса
                        ; прерывания 13h
        pop     ES
        pop     DS

        push    ES      ; Подготовка и
        mov     AX,DS:[2ch] ; освобождение DOS
        mov     ES,AX   ; environment
        mov     AH,49h  ; для уменьшения
        int     21h     ; размера резидентной
                        ; части программы
        pop     ES
        push    DS
        push    ES
        mov     AX,3513h ; Запомнить адрес
                        ; прерывания 13h
        int     21h
        mov     CS:old13o,BX
        mov     CS:old13s,ES

        cli             ; Запретить прерывания

        mov     AX,CS
        mov     DS,AX
        lea     DX,int13h
        mov     AX,2513h ; Установка нового адреса
        int     21h     ; прерывания 13h

        sti             ; Конец блока настройки DOS,
                        ; разрешить прерывания
        pop     ES
        pop     DS
        mov     DX,offset install; Установить
        int     27h     ; резидентную часть
                        ; программы
already_load:
        pop     ES
        pop     DS
        ret

install endp

code    ends
        end     start
```

```
; Программа переключения комплекса PROTECT из
; активного состояния защиты в пассивное и наоборот.
; Проверяет первые 6 байт кода прерывания 13h. Если
; они совпадают с кодом в PROTECT, то производится
; переключение ключевого байта из одного состояния в
; другое.
; This program was written by JVK Software.
; All rights reserved. Декабрь 1990
;
; Эта программа является freeware и может
; передаваться в неизмененном виде всем
; пользователям в некоммерческих целях.
; Автор Ю. Кравацкий

code    segment para public
        assume  CS:code,DS:code

        org     100h

start:
        push    ES
        mov     AX,3513h
        int     21h     ; Получить адрес
                        ; прерывания 13h
        mov     AX,word ptr ES:BX
        cmp     AX,080FAh ; Проверка первого слова
        jne     no_protect ; кода протектора

        inc     BX
        inc     BX
```

```
        mov     AX,word ptr ES:BX
        cmp     AX,003FCh ; Проверка второго слова
        jne     no_protect  ; кода протектора

        inc     BX
        inc     BX
        mov     AX,word ptr ES:BX
        cmp     AX,00C74h ; Проверка третьего
        jne     no_protect  ; слова кода протектора

        mov     BX,0117h    ; Загрузить в BX смещение
                            ; ключевого байта
        mov     AH,byte ptr ES:BX
        cmp     AH,075h   ; Защита записи включена?
        je      unprotect
        cmp     AH,0EBh   ; Защита записи выключена?
        je      protect
        jmp     bad_protect ; Несогласованные версии
                            ; PROTECT и PROTOFF
unprotect:
        mov     AH,0EBh              ; Установить
        mov     byte ptr ES:BX,AH; ключевой байт
                                 ; в состояние
                                 ; разрешения записи
        lea     DX,unpro_message ; Выдать сообщение о
        mov     AX,0900h         ; разрешении записи
        int     21h
        jmp     exit

protect:
        mov     AH,075h              ; Установить
        mov     byte ptr ES:BX,AH; ключевой байт
                                 ; в состояние запрета
```

```
                                        ; записи
        lea     DX,prot_message   ; Выдать сообщение о
        mov     AX,0900h          ; запрете записи
        int     21h
        jmp     exit

no_protect:
        lea     DX,no_prot_mess   ; Выдать сообщение о
        mov     AX,0900h          ; том, что
        int     21h               ; резидентная часть
                                  ; PROTECT в памяти
                                  ; не обнаружена
        jmp     exit

bad_protect:
        lea     DX,bad_message    ; Выдать сообщение о
        mov     AX,0900h          ; несоответствии
        int     21h               ; версий PROTECT и
                                  ; PROTOFF
exit:
        pop     ES
        ret

no_prot_mess   db  'PROTECT not found!', 0Dh, 0Ah, '$'
bad_message    db  'Unknown PROTECT version!'
               db  0Dh, 0Ah, '$'
unpro_messagedb    'Hard disk protection is now OFF'
               db  0Dh, 0Ah, '$'
prot_message   db  'Hard disk protection is now ON'
               db  0Dh, 0Ah, '$'

        ends
        end     start
```

Программная защита гибких дисков

В некоторых случаях возникает необходимость программной защиты от записи также и дисководов гибких дисков. Это может понадобиться, например, если вы по каким-либо причинам не хотите “заклеивать“ дискету, но хотите вместе с тем иметь гарантию, что на нее не будет произведена никакая запись. Помимо этого иногда бывает необходимо полностью запретить обращение к флоппи-дисководам, например, для защиты их от несанкционированного доступа. Последняя операция имеет смысл только в том случае, когда нет возможности перезагрузить компьютер с дискеты (на некоторых компьютерах это предусмотрено аппаратно). Однако не слишком квалифицированных пользователей подобная мера обычно все равно приводит в полнейшее замешательство и отбивает всякую охоту к дальнейшим экспериментам.

Предлагаемая программа FLPT (FloppyProtect) позволяет выполнять указанные выше действия. Имеются три режима работы программы — с защитой флоппи -дисководов только от записи, полным запретом обращения к флоппи-дисководам и режим отключенной защиты, в котором чтение и запись разрешены.

Переключение режимов защиты осуществляется вызовом той же программы FloppyProtect с соответствующей опцией в командной строке. Список возможных опций имеется в тексте программы. Если была задана неверная опция, либо если программа была запущена вообще без каких-либо ключей, то на экран будет выведен список разрешенных опций.

При каждом запуске программы вначале осуществляется проверка наличия в памяти ее резидентной части. Таким образом, резидентная часть программы может быть загружена только один раз, а при последующих запусках будет производиться лишь переключение режимов защиты.

Для отслеживания в памяти резидентной части программы, а также для переключения режимов защиты используется нестандартная функция 77h прерывания 13h (номер функции выбран произвольно). В программе имеются специальные участки кода, надлежащим образом реагирующие на вызов функции 13h, 77h. При этом выполняются нужные переключения и осуществляется возврат необходимых значений.

Благодаря такому техническому приему, программа FloppyProtect становится независимой от работы других резидентных программ, загружаемых в память после нее и тоже использующих прерывание 13h.

Следует иметь в виду, что программа FloppyProtect устанавливает защиту сразу на все флоппи-дисководы, подключенные к вашей системе. Если вы хотите иметь возможность устанавливать защиту избирательно — измените соответствующим образом нужные участки программы.

А. Синев

```
; Программа FloppyProtect. Файл FLPT.ASM
; Программа устанавливает режимы запрета записи или
; чтения/записи на floppy-дисководе.
; Автор А. Синев

code        segment  byte public
            assume   cs:code, ds:code

            org      100h

flpt        proc     far
start:
            jmp      short initialize    ; Инициализация
flpt        endp

; Точка входа в программу по прерыванию 13h
int_13h     proc     far
            cmp      ah,77h         ; Проверка нестандартной
            jne      chkptmode      ; функции 13h, 77h
            mov      byte ptr cs:ptmode,bl; Установить новый
                                    ; режим защиты
            mov      ah,78h         ; Значение, возвращаемое
                                    ; функцией 13h, 77h
            retf     2              ; Возврат с увеличением указателя
                                    ; стека на 2 для удаления слова
                                    ; состояния флагов, засланного
                                    ; при вызове прерывания 13h
int_13h     endp

ptmode      db       0              ; Ключевой байт текущего режима
                                    ; защиты
chkptmode:
            cmp      dl,80h         ; Проверка обращения к floppy-
                                    ; дисководу (номера < 80h)
            jae      old_int13h     ; Если обращение к жесткому
                                    ; диску, то переход на вызов
                                    ; старой программы обработки
                                    ; прерывания 13h
            cmp      byte ptr cs:ptmode,'N' ; Проверка отключенной
                                    ; защиты
            je       old_int13h
            cmp      byte ptr cs:ptmode,'W' ; Проверка режима
                                    ; запрета записи
            jne      set_errstat    ; Если режим запрета
                                    ; чтения/записи
            cmp      ah,2           ; Проверка обращения к
                                    ; функциям чтения
            jbe      old_int13h
set_errstat:
            stc                     ; Установить Carry-флаг для
                                    ; индикации ошибки
            mov      ah,80h         ; Установить код ошибки
                                    ; (Drive not ready)
            retf     2              ; Возврат
old_int13h:                         ; Вызов старой программы
                                    ; обработки прерывания 13h
            db       0EAh           ; Первый байт инструкции
                                    ; far jump
offst       dw       0              ; Смещение
segmnt      dw       0              ; Сегмент

initialize:                         ; Программа инициализации
            mov      bl,ds:[5Dh]    ; FCB1[1] — адрес имени
                                    ; файла (первого параметра из
                                    ; командной строки) в PSP
            and      bl,11011111b   ; Преобразовать символ в
                                    ; верхний регистр
            mov      switch,bl      ; Временно сохранить
            cmp      bl,'W'         ; Анализ опции
            mov      dx,offset wpt_mes    ; Адрес сообщения
                                    ; о запрете записи
            je       chkint13h
            cmp      bl,'R'
            mov      dx,offset rwpt_mes   ; Адрес сообщения о
                                    ; запрете чтения/записи
            je       chkint13h
            cmp      bl,'N'
            mov      dx,offset rwen_mes   ; Адрес сообщения об
                                    ; отключении защиты
                                    ; (разрешении
                                    ; чтения/записи)
            je       chkint13h

            mov      ah,9           ; Вывести сообщение о
            mov      dx,offset illsynt_mes ; непредусмотренной
            int      21h            ; опции в командной
                                    ; строке
            int      20h            ; Выход из программы
chkint13h:
            mov      ah,9           ; Функция DOS 9h "Print
            int      21h            ; string at ds:dx"

            xor      dx,dx          ; Обращение к floppy-дисководу
                                    ; (очистка dl необходима для
                                    ; корректной работы
                                    ; FloppyProtect в окружении
                                    ; других резидентных
                                    ; программ, также
                                    ; использующих
                                    ; прерывание 13h)
            mov      ah,77h         ; Вызов нестандартной
            int      13h            ; функции 13h, 77h
            cmp      ah,78h         ; Проверка возвращаемого
                                    ; значения
            jne      setnewint13h   ; Если не соответствует, то
                                    ; установить новый вектор
                                    ; прерывания 13h
            mov      ah,9           ; иначе — вывести
            mov      dx,offset ptld_mes; сообщение о том, что
            int      21h            ; резидентная часть
                                    ; программы уже загружена
            int      20h            ; Выход из программы
setnewint13h:
            mov      ax,3513h       ; Получить старый
                                    ; вектор прерывания 13h
            int      21h            ; Функция DOS 35h "Get
                                    ; interrupt vector in es:bx"
            mov      offst,bx       ; Сохранить старый
            mov      segmnt,es      ; вектор прерывания 13h

            mov      ax,2513h       ; Установить новый вектор
            mov      dx,offset int_13h    ; прерывания 13h
            int      21h            ; Функция DOS 25h "Set
                                    ; interrupt vector to ds:dx"
            mov      ah,9           ; Вывести сообщение об
            mov      dx,offset inst_mes   ; установленной программе
            int      21h

            mov      bl,switch      ; Загрузить опцию
                                    ; переключения режима
                                    ; защиты
            xor      dx,dx          ; Обращение к floppy-
```

```
                                        ; дисководу
        mov  ah,77h                     ; Вызвать нестандартную
        int  13h                        ; функцию 13h, 77h
                                        ; переключения режима
        mov  dx,offset initialize       ; Выйти из программы,
        int  27h                        ; оставив в памяти ее
                                        ; резидентную часть

switch  db   0                  ; Переменная для временного
                                ; хранения переключателя
                                ; режима защиты
inst_mes db  'FloppyProtect is now installed.', 0Dh, 0Ah, '$'
ptld_mes db  'FloppyProtect was already loaded.'
         db  0Dh, 0Ah, '$'
wpt_mes  db  'The floppy drive is write prohibited.'
         db  0Dh, 0Ah, '$'
rwpt_mes db  'The floppy drive is read and write prohibited'
         db  0Dh, 0Ah, '$'
rwen_mes db  'The floppy drive is read and write enabled.'
         db  0Dh, 0Ah, '$'
illsynt_mes db 'Syntax:  FLPT /[w,r,n]', 0Dh, 0Ah
         db  '/w = write protect', 0Dh, 0Ah
         db  '/r = read and write protect', 0Dh, 0Ah
         db  '/n = no protection', 0Dh, 0Ah, '$'

code     ends

         end  start
```

Язык логического программирования МПРОЛОГ

1. Что такое МПРОЛОГ?

МПРОЛОГ — язык модульного логического программирования. Название МПРОЛОГ образовано от слов Модульное ПРОграммирование ЛОГическое.

Язык МПРОЛОГ предназначен для решения задач, относящихся к проблематике искусственного интеллекта. К числу таких задач относятся создание экспертных систем и инструментальных средств для их разработки, систем автоматизированного проектирования с элементами искусственного интеллекта, создание интеллектуальных баз данных, автоматическое доказательство теорем и т.п.

В настоящее время язык МПРОЛОГ уже получил очень широкое распространение для решения различных задач в таких областях, как машиностроение, приборостроение, химия, медицина, архитектура и т.д.

На языке МПРОЛОГ создан ряд экспертных систем в различных прикладных областях.

2. В чем отличие МПРОЛОГа от алгоритмических языков ?

Язык МПРОЛОГ принципиально отличается от традиционных алгоритмических языков программирования тем, что в нем требуется описывать логическую модель предметной области решаемой задачи в терминах объектов, их свойств и отношений между объектами без подробного описания самого алгоритма решения задачи.

Программа на языке МПРОЛОГ состоит из множества утверждений. Каждое из утверждений является либо некоторым фактом о заданной предметной области, либо правилом, указывающим, как решение связано с имеющимися в программе фактами или каким образом это решение можно получить из имеющихся фактов.

В основу языка МПРОЛОГ положены идеи логического программирования.

3. В чем особенности языка МПРОЛОГ ?

Язык МПРОЛОГ является одной из наиболее известных и широко распространенных версий языка логического программирования Пролог.

Основные характерные черты языка МПРОЛОГ :

— возможность использования концепций модульного логического программирования при создании программ, что позволяет создавать большие программы, состоящие из модулей, а также вести разработку больших программных систем по частям и т.п.;

— большой набор встроенных предикатов (более 150) и операторов (более 50), выполняющих различные функции и существенно облегчающих программирование на языке МПРОЛОГ ;

— наличие набора функций трехмерной машинной графики;

— возможность программирования пользователем реакции системы на ошибки, возникающие во время выполнения его программ;

— наличие интерфейса с традиционными процедурными языками программирования, такими как Си, Паскаль, Фортран, Кобол, Ассемблер, что значительно расширяет функциональные возможности системы и повышает ее эффективность;

— совместимость программ, написанных на языке МПРОЛОГ, для различных типов ЭВМ;

— наличие мощной и удобной системы поддержки разработки программ на языке МПРОЛОГ, включающей в себя редактор текстов и отладчик;

— возможность создания больших программ благодаря тому, что интерпретатор имеет большие таблицы (т.е. фактически не накладываются ограничения на размер программ, допускается использование имен переменных любой длины);

— эффективный по времени и памяти механизм выполнения программ, заложенный в интерпретаторе и включающий в себя, например, такие средства, как хвостовая рекурсия, управление поиском и т.п.;

— допустимость компиляции и оптимизации разрабатываемых на языке МПРОЛОГ программ с целью повышения их эффективности по времени и занимаемой памяти ;

— возможность создания проблемно-ориентированных расширений системы МПРОЛОГ, например, таких как система Т-ПРОЛОГ, реализованная на языке МПРОЛОГ, которая предназначена для решения задач моделирования систем с дискретными событиями.

4. Где используется язык МПРОЛОГ ?

Первый интерпретатор языка Пролог был создан в 1972 г. в Марселе. В 70-е годы Прологом интересовался очень узкий круг специалистов, главным образом из академических и университетских кругов.

В конце 70-х годов язык Пролог был выбран в качестве языка-ядра в японском Проекте компьютеров пятого поколения. Этот факт, а также широкое развитие работ в области искусственного интеллекта, в частности по созданию экспертных систем, в значительной степени способствовало росту популярности языка и его активному использованию.

В 80-е годы Пролог стали широко использовать для решения широкого круга прикладных задач в самых разнообразных областях. Во второй половине 80-х годов началось использование языка Пролог в ведущих научно-исследовательских организациях СССР.

В настоящее время система МПРОЛОГ в СССР используется более чем в 170 организациях. География использования системы МПРОЛОГ насчитывает свыше 70 городов. Система МПРОЛОГ распространяется как коммерческий продукт более чем в 10 странах мира, включая США, Японию, ФРГ, Канаду и т.д. К настоящему времени продано несколько тысяч экземпляров системы.

Использование системы МПРОЛОГ наиболее эффективно в тех прикладных областях, в которых требуется:

1) Создание экспертных систем.

МПРОЛОГ является "естественным" языком для создания экспертных систем. Знания эксперта в конкретной области могут быть очень эффективно представлены в виде Пролог-программы. Основная работа экспертной системы состоит в выполнении логического вывода на основании фактов и правил, относящихся к области данной задачи.

2) Получение ответов на вопросы с помощью логического вывода (автоматическое доказательство теорем, синтез и верификация программ, вопросно-ответные системы и т.п.).

3) Выполнение символьных вычислений (например, обработка текстов естественного языка). Возможности символьных вычислений в языке МПРОЛОГ сопоставимы с соответствующими возможностями языка Лисп, а в сочетании с другими средствами языка МПРОЛОГ — даже и превосходят их.

4) Поиск в базе данных, содержащей не только факты, но и правила, формулирующие законы о взаимоотношениях и взаимозависимости этих фактов. Типичная ситуация, подходящая для применения языка МПРОЛОГ — это наличие некоторого набора условий, которым должен удовлетворять объект.

Программу на языке МПРОЛОГ с некоторыми допущениями можно рассматривать как реляционную базу данных, которая содержит отношения, выражающие факты и правила, и позволяет делать запросы и добавлять новую информацию в базу данных. Утверждение может быть просто добавлено как новое правило языка МПРОЛОГ. К реляционной базе данных (т.е. к программе) допустимы самые разнообразные запросы, но такие, ответ на которые может быть логически выведен системой МПРОЛОГ из содержимого базы данных.

При организации хранения информации не надо придерживаться какой-нибудь заранее принятой схемы. Это относится только к той части базы данных, которая находится в оперативной памяти компьютера и непосредственно обрабатывается программой на языке МПРОЛОГ. Когда надо обрабатывать большие объемы данных, система МПРОЛОГ должна быть подходящим образом связана с системой управления базой данных (СУБД).

5) Конструирование объектов, удовлетворяющих определенному набору условий (автоматизированное проектирование в области архитектуры, химии, микроэлектроники и т.п.).

6) Имитация реальных ситуаций, включающих параллельные процессы, определенные непроцедурным способом, т.е. моделирование высокого уровня, когда свойства модели можно описать утверждениями на языке МПРОЛОГ.

7) Быстрая разработка на МПРОЛОГе небольших по объему прототипов больших и сложных систем в процессе создания последних, например, в тех случаях, когда важно за короткое время перейти от спецификации задачи к работающей программе.

5. Пример элементарной программы на языке МПРОЛОГ

Рассмотрим простейшую программу на языке МПРОЛОГ. Программа содержит информацию о родственных отношениях в семье.

Программа состоит из одного модуля по имени "семья", в котором находятся восемь фактов, три правила и комментарии. Факты содержат информацию о родственных отношениях в семье. Пояснения к тексту программы оформлены в виде комментариев на языке МПРОЛОГ. Ниже приведен текст программы.

```
module семья.   — Заголовок модуля по имени семья.
/* Пример программы на языке МПРОЛОГ*/
/*$eject*/   — Служебное слово, сообщающее
               претранслятору о необходимости начать
               новую страницу в листинге при претрансляции
body.          — Начало тела модуля

/* Комментарий к определению "отец"          */
```

```
/* Информация об отцах :                          */
/* Факты вида "отец(X,Y)" означают,               */
/* что X является отцом Y                         */
/* Обратите внимание, что собственные имена       */
/* пишутся с маленькой                            */
/* буквы.                                         */

отец(иван,федор).
отец(иван,николай).
отец(иван,максим).
отец(федор,виктор).
отец(федор,степан).
отец(николай,сидор).

endmod /* семья */.              — Признак конца модуля
```

К приведенной выше программе можно обратиться с различными вопросами (запросами).

Примечание.
Если у Вас есть возможность работать с системой МПРОЛОГ, то рекомендуем Вам в процессе чтения настоящей статьи выполнять примеры программ на компьютере.

Зададим программе вопрос, является ли Иван отцом Федора:

?отец(иван,федор).

Интерпретатор сопоставит полученный вопрос с имеющимися в программе фактами (т.е. с содержимым внутренней базы данных).

Сопоставление закончится успешно, т.к. вопрос в точности совпал с одним из фактов программы.

На экран будет выдан ответ "Yes" (да), и вопрос о необходимости продолжения поиска новых вариантов решений:

Continue (y/n) ? (Продолжить ли (да/нет) ?)

Ответим системе "n", т.к. мы знаем, что в программе содержится всего один подходящий факт, следовательно, поиск других вариантов решений не имеет смысла (в виду их отсутствия).

Зададим программе вопрос, является ли Иван отцом Виктора:

?отец(иван,виктор).

Программа даст отрицательный ответ "NO" (нет), т.к. Иван не является отцом Виктора.

Зададим программе вопрос, является ли Михаил отцом Василия:

?отец(михаил,василий).

Программа даст такой же отрицательный ответ "NO", но в этот ответ вложен скорее смысл "не известно" или "нет данных", т.к. в нашей программе нет информации о родственных отношениях Михаила и Василия.

В двух последних случаях мы получили одинаковые ответы "NO". Но между этими двумя, внешне одинаковыми ответами, имеется большое содержательное отличие. В первом случае полученный ответ может трактоваться как отрицательный ответ "нет",т.е. указанные в нашей базе данных Иван и Виктор не являются отцом и сыном, т.е. содержание запроса противоречит содержанию базы данных программы, а во втором случае ответ может трактоваться как "не известно", "нет данных", "не знаю" и т.п., потому что у программы просто нет данных, не хватает информации для того, чтобы дать правильный ответ на поставленный вопрос.

Теперь рассмотрим вопросы, в которых используются переменные. Переменные в языке МПРОЛОГ могут начинаться с прописной латинской буквы или с символа подчеркивания.

Определим, чьим отцом является Иван, обозначив переменную прописной латинской буквой, и задав программе вопрос :

?отец(иван,X).

Программа ответит: X = федор и предложит продолжить поиск других возможных решений, задав нам вопрос: Continue (y/n)?.

Если мы будем каждый раз отвечать y (т.е. "да"), то программа найдет все возможные варианты решений:

X = николай, X = максим.

Этот вопрос можно представить и по-другому, используя второй вариант обозначения переменных, с помощью знака подчеркивания. Тогда переменные могут быть представлены в более наглядной форме, например, в виде "_сын_ивана".

?отец(иван,_сын_ивана).

Программа ответит: сын_ивана = федор и, как и в предыдущем случае, запросит нас о том, надо ли продолжить поиск других ответов: Continue (y/n)?

Ответив "y" или просто нажав клавишу ввода «Enter», мы получим следующий вариант ответа: сын_ивана = николай. Поиск и вывод на экран новых ответов будет продолжаться до тех пор, пока не исчерпаются все подходящие данные (факты), имеющиеся в базе данных программы. Если мы ответим "n", то программа прекратит дальнейший поиск вариантов ответов.

В случае, если в программе присутствует несколько одинаковых имен, программа будет их находить и выводить на экран столько раз, сколько раз они встречаются в программе.

Например, рассмотрим случай, когда в семье дед, отец, внук и правнук имеют одинаковые имена.

```
отец(иван,иван).
отец(иван,иван).
отец(иван,иван).
```

В ответ на запрос

?отец(иван,X).

мы можем трижды получить один и тот же ответ: X = иван.

Точнее говоря, мы получим три одинаковых ответа, обозначающих отца, внука и правнука соответственно.

Найдем имеющиеся в базе данных пары "отец-сын", задав вопрос:

?отец(X,Y).

или, что более наглядно:

?отец(_отец,_сын).

В первом случае программа ответит:

```
X = иван
Y = федор
```

Во втором:

```
отец = иван
сын = федор
```

и запросит о необходимости продолжения поиска других вариантов ответов: Continue (y/n)?

Отвечая "у", или просто нажимая клавишу ввода «Enter», мы получим все возможные варианты ответов, т.е. все имеющиеся в базе данных программы пары "отец-сын".

Теперь рассмотрим на языке МПРОЛОГ правило, определяющее, что значит быть братом: два человека являются братьями, если имеют общего отца. На языке МПРОЛОГ это утверждение может быть представлено следующим образом:

```
брат(X,Y) :-
    отец(Z,X), отец(Z,Y).
```

Это правило обладает одним существенным недостатком — с его помощью будут найдены люди, сами себе являющиеся братьями.

Введем условие, отсекающее подобные решения, т.е. условие типа "сам себе не брат" (not X = Y), тогда правило будет иметь вид:

```
брат(X,Y) :-
    отец(Z,X), отец(Z,Y), not X = Y.
```

Определим с помощью этого правила, являются ли братьями Федор и Николай, Федор и Сидор. Для этого зададим системе следующие вопросы :

```
?брат(федор,николай).
?брат(федор,сидор).
```

Затем определим братьев Федора:

```
?брат(федор,_брат_федора).
```

Определим все пары братьев:

```
?брат(X,Y).
```

Теперь найдем человека, имеющего брата. Причем, нас интересует только человек, имеющий брата. Имя брата нас не интересует. В этом случае для обозначения имени брата можно использовать безымянную переменную, обозначаемую символом подчеркивания. Запрос будет иметь вид:

```
?брат(X,_).
```

Рассмотрим правило, определяющее, что значит быть дедом: дед — это отец отца.

```
дед(X,Y) :-
    отец(X,Z), отец(Z,Y).
```

Рассмотрим другие, более сложные правила, определяющие, что значит быть дедом: дед — это отец отца или отец матери. Такое определение состоит уже из двух предложений, первое из которых определяет деда по отцу, а второе — деда по матери.

```
дед(X,Y) :-
    отец(X,Z), отец(Z,Y).
дед(X,Y) :-
    отец(X,Z), мать(Z,Y).
```

Это определение можно записать короче, в виде одного предложения, если использовать логическую связку "или", обозначаемую в языке МПРОЛОГ точкой с запятой " ; ". Тогда правило, определяющее деда, будет выглядеть следующим образом:

```
дед(X,Y) :-
    отец(X,Z), отец(Z,Y) ;
    отец(X,Z), мать(Z,Y).
```

Запись может быть сделана еще короче, если использовать скобки:

```
дед(X,Y) :-
отец(X,Z), ( отец(Z,Y) ; мать(Z,Y) ).
```

С помощью этого правила определим, чьим дедом является Иван. Определим всех внуков Ивана. Рассмотрим правило, определяющее что значит быть родителем: родитель — это отец или мать.

```
родитель(X,Y) :-
    отец(X,Y);
    мать(X,Y).
```

Определим родителей Федора и Виктора: ?родитель(X,федор). После введения правила, определяющего родителя, правило, определяющее деда, формулируется так : дед — это отец родителя:

```
дед(X,Y) :-
    отец(X,Z), родитель(Z,Y).
```

Программу можно усложнить еще, добавив в качестве аргументов, например, год рождения. Программа будет иметь следующий вид:

```
отец(иван,1925,николай,1960).
отец(иван,1925,константин,1970).
отец(иван,1925,федор,1980).
```

Тогда в запрос можно включать год рождения, например: найти сыновей Ивана, родившихся до 1965 года :

```
?отец(иван,_,X,Y),Y<1965.
```

И наконец, введем правило, определяющее, что значит быть сыном своих родителей:

сын(X,Y,Z) — будет означать, что X является сыном своего отца Y и матери Z. Тогда правило имеет вид:

```
сын(X,Y,Z):-
    отец(Y,X),мать(Z,X).
```

С помощью этого правила мы можем определить, например, сына Ивана и Марии, задав вопрос:

```
?сын(X,иван,мария).
```

Аналогично можно определить, кто является сыном Анастасии, причем, кто является отцом, нас не интересует, поэтому для обозначения отца используем безымянную переменную :

```
?сын(X,_,анастасия).
```

Аналогично можно определить имя сына, не интересуясь именами его родителей. Обозначим имена родителей безымянными переменными, тогда вопрос будет иметь следующий вид :

```
?сын(X,_,_).
```

В дальнейшем мы рассмотрим более сложные примеры программ на языке МПРОЛОГ.

И в заключение, хотелось бы предложить ответы еще на три вопроса.

6. На каком языке реализована система МПРОЛОГ?

Система МПРОЛОГ реализована частично на ассемблере, частично на самом языке МПРОЛОГ, частично на языке описания компиляторов CDL-2.

7.Кто является официальным дистрибьютором системы МПРОЛОГ?

Система МПРОЛОГ официально распространяется на территории СССР Институтом проблем информатики Академии Наук СССР через малое предприятие "Информатика", учредителем которого является ИПИ АН СССР.

Система МПРОЛОГ распространяется для следующих ЭВМ и операционных систем (табл.1).

Таблица 1

Тип ЭВМ	Операционная система
IBM PC	MS DOS, PC DOS
ЕС	СВМ/ПДО, VM/CMS
VAX	ИНМОС, МОС ВП UNIX, VMS

8. Как получить систему МПРОЛОГ ?

Заявки на получение системы МПРОЛОГ с указанием вида ЭВМ и операционной системы необходимо направлять в малое предприятие "Информатика" по адресу:

117900, Москва, ГСП-1, В-334, ул. Вавилова,
дом 30/6, ИПИ АН СССР, МП "Информатика".
Справки по тел.: (095) 362-46-54.
Факс: (095) 310-70-50. Телекс 411853 INFO SU.

Ю.Тихонов

Литература:

1. Иванова Г.С., Тихонов Ю.В. "Введение в язык МПРОЛОГ", М.: Издательство МГТУ им. Н.Э. Баумана, 1990 г. — 152 с.
2. Калиниченко Л.А.,Степанов А.И.,Тихонов Ю.В. "Система МПРОЛОГ для автоматизации обработки знаний на ЭВМ." Серия "Методические материалы и документация по пакетам прикладных программ". Выпуск 59. М.:МЦНТИ, 1989. — 112 с.
3. Клоксин У., Меллиш К. "Программирование на языке Пролог", М.: Мир, 1987. — 336 с.

Фирма-производитель жестких дисков Micropolis начала выпуск дисков, предназначенных для высокопроизводительных персональных компьютеров. В контроллерах дисков использованы алгоритмы кэширования, подобные применяемым на суперкомпьютерах, которые позволяют уменьшить среднее время доступа до 3.9 миллисекунд. Такая скорость доступа была достигнута при использовании встроенной кэш-памяти объемом от 64 до 256 Кбайт, в зависимости от емкости диска.

Micropolis планирует также выход на рынок дисковых массивов с технологией, использующей уже имеющиеся дисковые контроллеры на базе 386 и 486 процессоров для выработки контрольного разряда, обеспечивающего проверку и восстановление целостности данных.

Electronic Engineering Times, November 5, 1990.

Во втором квартале 1991 года объемы продажи компьютеров Macintosh фирмы Apple выросли на 85 % по сравнению с аналогичным периодом прошлого года. Прибыли компании выросли на 19%, а дивиденды по акциям — на 3%.

За предыдущий квартал, окончившийся 21 марта 1991 года, фирма продала компьютеров на 1598 миллиарда долл. Все больше и больше машин (52%) реализуется за пределами США.

"Возрастающий спрос на новые недорогие модели вызвал новый период подъема компании как на американском, так и на международном рынках", — сказал президент фирмы Джон Скали.

Newsbytes News Network, April 15, 1991.

На выставке Comtek'91 в Москве фирма Samsung впервые представила свою новую марку для СССР и Восточной Европы — Syntronix. Под этой маркой будут продаваться сделанные в США компьютеры на базе i286 и i386, факсы, телевизоры, видеокамеры и телефоны.

Компьютеры будут поставляться с прошитой кириллицей в видеоадаптерах и стандартными русско-латинскими клавиатурами.

Что еще нужно для советского рынка? Надежность! Один из компьютеров (вероятно, специально подготовленный, но тем не менее) проработал неделю на выставке в наполненном водой аквариуме.

Фирма ComputerLand, имеющая сейчас магазин в Москве и филиалы в Киеве и Ленинграде, первой подписала контракт на продажу этих машин в СССР.

Newsbytes News Network, April 15, 1991.

Парад СУБД завершен. Но... парад продолжается. Если вы захотите узнать о других программных продуктах, предназначенных для управления базами данных, пишите нам, мы с удовольствием ответим.

Парад СУБД продолжается...

Фирма Oracle
Программа Oracle 5.1.b

Системные требования

Программа Oracle функционирует на компьютерах с процессором 80286 или 80386, имеющих не менее 640 Кбайтов оперативной памяти и 2.5 Мбайта расширенной памяти, один дисковод, 10 Мбайтов памяти на жестком диске и операционную систему DOS версии 2.1 и выше.

Характеристика программы

Появление операционной системы OS/2 и стандартного встроенного UNIX для PC вызвало повышенный интерес к программам, разработанным для гетерогенных распределенных систем. В области баз данных этот интерес выразился во включении новых функций, обеспечивающих многозадачность, поддержку языка SQL и совместимость с базами данных универсальных компьютеров. Oracle, несмотря на высокую сложность освоения этого пакета, содержит в себе все необходимое для работы в распределенных системах.

Вследствие высокой мощности пакета и его прототипов, работающих на универсальных компьютерах, Oracle зарекомендовал себя превосходным средством разработки больших баз данных, особенно в тех случаях, когда микрокомпьютеры составляют лишь часть вычислительных мощностей организации.

Что касается конкуренции, то Oracle испытывает ее прежде всего со стороны пакетов, также функционирующих на многих платформах, и прежде всего PC Focus, Ingres и Informix. Совместные разработки, ведущиеся Digital Equipment, Relational Technology, Ashton-Tate и Cullet, в части, касающейся создания совместимой с SQL базы данных Rdb для операционной системы VAX/VMS, также могут иметь серьезные последствия для Oracle, занимающего сейчас весьма значительную долю рынка СУБД для VAX. Наконец, сервер SQL фирм Ashton-Tate, Microsoft и Sybase должен обеспечить интерфейс с базами данных в локальных сетях.

Oracle является модульной системой, а каждый модуль представляет собой отдельную программу со своими характеристиками и требованиями. Это накладывает свой отпечаток и на пользовательский интерфейс. Практически в каждом из шести модулей Oracle собственный интерфейс, а из меню одного модуля невозможно обратиться к другому. Назначение функциональных клавиш в разных модулях также отличается. Некоторые модули Oracle имеют систему меню (к ним относятся SQL*ReportWriter, SQL*Forms, SQL*Calc), другие работают лишь в командном режиме (SQL*Plus).

Проектирование базы данных в Oracle возможно двумя способами. В SQL*Forms создание базы данных осуществляется по системе подсказок, в SQL*Plus используется команда create table.

При проектировании форм документов в Oracle используется меню-ориетированная программа SQL*ReportWriter. Модуль на каждом из этапов ввода высвечивает на экране соответствующие подсказки. Одновременно с формированием отчета ReportWriter формирует соответствующие операторы SQL. Модуль позволяет создавать отчеты в широком диапазоне, начиная с простых бланков, определяемых по умолчанию, и заканчивая пользовательскими отчетами, имеющими колонтитулы и другие сложные элементы.

Модуль SQL*Forms предоставляет меню-ориентированный оконный интерфейс для создания экранных форм базы данных. Меню отражает практически все возможные варианты создания баз данных. Единственное исключение составляют индексы, которые определяются модулем SQL*Plus после создания таблицы.

Вводить данные можно либо с помощью команды insert в SQL*Plus, либо в SQL*Forms.

Запросы в Oracle также обрабатываются двумя различными способами. SQL*Plus поддерживает стандартный язык запросов SQL, а SQL*Forms имеет функцию

Query-By-Forms, когда запрос формируется в шаблоне с указанием имен и логических операций.

Oracle имеет чрезвычайно мощные реляционные средства и средства программирования. Размер и сложность базы данных в большинстве своем ограничены только возможностями операционной системы и объемами доступной памяти. Oracle поддерживает связи между записями типа "один-ко-многим", "многие-к-одному", "многие-ко-многим". Мощные поисковые средства позволяют задавать альтернативные критерии, включать в условия поиска отрицание и диапазоны значений.

Oracle позволяет динамически изменять структуру базы данных.

SQL Oracle обеспечивает полную совместимость с языками IBM SQL/DS и DB/2.

Структура данных

Число таблиц, с которыми может одновременно работать Oracle, не ограничено. Файл может включать до 255 полей, но не более 64.000 байтов в записи. Oracle 5.1.b поддерживает символьные, текстовые (длиной до 64 Кбайтов), числовые и долларовые данные, а также форматы даты и времени. С помощью триггеров можно определить данные типа "да/нет" и константы.

Несмотря на то, что Oracle имеет широкие возможности экспорта и импорта файлов в форматах баз данных больших машин, он не поддерживает ни одного из форматов PC.

Защита данных

Oracle обладает, пожалуй, наиболее мощными средствами защиты данных из числа рассматриваемых. Программа имеет множество уровней регистрации и защиты данных в системе. Полномочия доступа к данным каждого из пользователей определяются только администратором базы данных.

Дополнительная информация

Однопользовательская версия Oracle Professional 5.1.b для DOS стоит 1299 долл. Продаются и версии для работы в локальной сети.

Предоставляются бесплатные телефонные консультации в течение 30-дневного гарантийного срока. Заключаются договоры на сопровождение. Полное сопровождение, включающее предоставление новых версий и телефонные консультации, стоит 499 долл. в год. В случае только предоставления новых версий сопровождение обойдется в 299 долл. в год.

Фирма Symantec
Программа Q&A версия 3.0

Системные требования

Программа Q&A работает на компьютерах IBM PC/XT/AT и PS/2 с оперативной памятью 512 Кбайт и операционной системой MS-DOS версии 2.0 и выше. Для работы с Q&A в локальной сети требуется 640 Кбайтов оперативной памяти и MS-DOS версии 4.0.

Характеристика программы

Q&A сочетает в себе возможности базы данных и текстового процессора. Она имеет также элементы искусственного интеллекта, позволяющие формулировать запросы на естественном (английском) языке. Программа значительно более мощна по сравнению с другими системами для обработки плоских файлов. В частности, имеются реляционные средства создания форм отчетов на основе нескольких файлов. Уровень возможностей Q&A позволяет отнести ее, наряду с RapidFile фирмы Ashton-Tate, Reflex фирмы Borland и PFS:Professional File фирмы Software Publishing, к категории простых программ обработки данных. Этот тип программ разработан для категории пользователей с потребностями, меньшими, чем у работающих, например, с dBASE.

Программа имеет систему меню, разработанную по аналогии с интерфейсом интегрированного семейства программ PFS.

Q&A включает пять интегрированных модулей: File, Report, Write, Intelligent Assistant и Utilities. Последний содержит опции для задания типа принтера, экспорта-импорта данных и выполнения из программы функций DOS.

Intelligent Assistant поставляется вместе со встроенным словарем, содержащим около 400 слов, объем словаря можно увеличивать. Программа позволяет формулировать запросы в виде предложений на английском языке.

Текстовый процессор Q&A имеет практически все необходимые средства, включая проверку орфографии, математические функции, шрифты и клавишные макрокоманды.

Создание базы данных, добавление и поиск данных в Q&A выполняется в меню File, при этом каждое действие сопровождается соответствующей подсказкой.

Средства программирования Q&A достаточно мощны для подобного рода системы. Допускается устанавливать значения полей, принимаемые по умолчанию, а также ограничивать их верхний и нижний пределы.

Помимо Intelligent Assistant, Q&A включает традиционные средства поиска по методу Query-by-Form. Имеются система шаблонов и средства поиска подобных слов.

В Q&A имеется команда XLookup, позволяющая создавать представления, содержащие поля нескольких файлов.

Генерация стандартных отчетов в Q&A не создает особых сложностей, однако включение дополнительных элементов требует использования кодов форматирования, что вызывает определенные сложности.

Существенным достоинством Q&A является многопользовательский режим работы. Синхронизация доступа обеспечивается путем блокировки записи первой поступившей на выполнение транзакции. Запись блокируется вплоть до выполнения запроса.

Структура данных

Поскольку Q&A не является полностью реляционной СУБД, она может одновременно работать только с одним файлом, однако, как отмечалось выше, имеются средства для создания представлений. Файл может включать до 2180 полей и иметь длину записи не более 64 Кбайтов. Q&A поддерживает 7 типов полей — текстовые (длиной не более 1638 байтов), числовые, долларовые, ключевые, логические, даты и времени.

В зависимости от длины полей допускается сортировка файла с вложенностью до 512 уровней. Кроме того, Q&A позволяет индексировать данные, число индексируемых полей в файле не превышает 115. Отметим, что другие аналогичные программы не имеют средств индексирования.

Q&A может импортировать файлы в форматах DIF, ASCII, Lotus 1-2-3, IBM Filing Assistant и dBASE II/III. Экспортировать файлы можно в формат DIF, ASCII переменной и фиксированной длины и dBASE II/III.

Защита данных

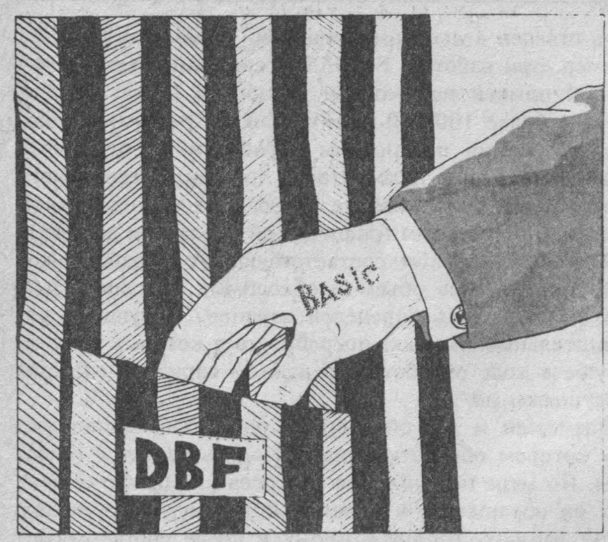
Q&A, в отличие от большинства программ подобного класса, имеет средства защиты данных. Уровни доступа устанавливаются администратором системы в экране Access Control Screen.

Дополнительная информация

Цена версии 3.0 Q&A составляет 349 долл. 5-дюймовые дискеты могут быть бесплатно заменены на 3-дюймовые, а дискеты для DOS на дискеты для OS/2. Замена дефектных дискет осуществляется бесплатно в течение 90-дневного гарантийного срока. Сетевое обеспечение Q&A Network Pack стоит 299 долл. Этот пакет приобретается для каждых трех пользователей. Зарегистрированные пользователи приобретают право на бесплатную подписку на ежеквартально издаваемый журнал и бесплатные телефонные консультации.

М.Михайлов

По материалам:
H.Edelstein, "An update on relational Technology", DataBased Advisor, June, 1990.
Datapro Reports on Microcomputers. Data Management.
G.Schussel, "The IBM Effect", DataBased Advisor, March, 1990.



ОПЫТ РАЗРАБОТКИ СПЕЦИАЛИЗИРОВАННЫХ БАЗ ДАННЫХ

Одним из основных направлений использования персональных ЭВМ является создание систем, основанных на применении разнообразных баз данных. Естественно, что в качестве инструментальных средств при этом используются в первую очередь системы управления базами данных, из которых сегодня наиболее популярны dBASE, FoxBASE, Clipper.

По такому пути пошли и мы, когда приступили к разработке специализированных систем накопления и обработки инженерно-геологических и гидрогеологических данных (базы фактографических данных). Использование языка FoxBASE позволило нам в минимальные сроки решить задачу первого этапа разработки — накопления данных: ввод, коррекцию, хранение, вывод информации.

Следует отметить, что проблема накопления данных имеет самостоятельную ценность для заказчика (а иногда и просто все его проблемы на этом заканчиваются). Поэтому важной задачей для нас являлась минимизация сроков разработки и внедрения данного варианта системы. В связи с этим при выборе конкретного инструмента разработки мы учитывали не только соответствие конкретного языка решаемой задаче, но и подготовленность специалистов-разработчиков. Так, при выборе FoxBASE мы исходили из того, что у нас накоплен значительный опыт разработки систем автоматизации бухгалтерского учета.

Однако в ходе дальнейшей работы по развитию системы стало очевидно, что в рамках только FoxBASE ее выполнить нельзя. Прежде всего это касалось обработки данных. Для их решения требуется активно использовать графические средства отображения и ввода данных, которые в стандартных СУБД практически отсутствуют. Кроме того, многие алгоритмы обработки (в частности, статистика, специальные системы обработки гидрогеологических данных) уже были реализованы ранее на других языках и перевод их на язык СУБД был нецелесообразен.

Второй проблемой явилась необходимость модульного построения системы. В частности, увеличение функциональных возможностей системы привело к резкому росту объема используемой оперативной памяти. Естественный шаг при решении этой проблемы — выделение подсистемы накопления и обработки данных в различные модули. В рамках FoxBASE создание такой модульной системы представляет определенные трудности.

С учетом этих аспектов было решено обрабатывающие программы писать в виде автономных модулей на языке QuickBASIC, который принят нами в качестве основного многофункционального инструментального средства разработки.

Таким образом, на данном этапе работы нами была принята двухуровневая организация системы: подсистема накопления данных, реализованная на FoxBASE, и подсистема обработки на QuickBASIC. Однако реализация такой системы потребовала обеспечения доступа к файлам базы данных со стороны обрабатывающих программ.

Традиционным решением этой задачи является организация обмена с файлами базы данных средствами СУБД, при этом взаимодействие с обрабатывающими программами производится через символьные последовательные данные (импорт исходных данных и эк-

спорт результатов обработки). Основной вариант схемы взаимодействия модулей системы выглядит следующим образом (рис. 1):

1. Головной модуль системы разрабатывается на FoxBASE. В нем находится главное меню системы, из которого можно попасть в головной раздел обработки данных, реализованный в рамках данного модуля.

2. Головной раздел обработки включает меню выполняемых операций, в соответствии с которыми производится чтение исходных данных из базы данных, их импорт в последовательный файл, запуск соответствующего обрабатывающего модуля, экспорт результатов и запись их в базу данных.

3. Обрабатывающий модуль (или набор модулей) реализуется на любом языке программирования. Обмен данными с базой данных (чтение исходной информации и вывод результатов, которые должны быть записаны в базу) производится через временные, создаваемые для выполнения конкретного вида обработки, последовательные файлы импорта-экспорта.

```
                                  ---------------------------------
                                 |   Головной  модуль системы      |
                                 |            FoxBASE              |
 -----------------               |=================================|
|  БАЗА ДАННЫХ    |              |   Подсистема накопления         |
|  -----------    |              |          данных                 |
|    файлы        | обмен данными|---------------------------------|
|  *.dbf, *.dbt   |=============>| Головной модуль подсистемы      |
|                 |<=============|        обработки:               |
 -----------------               | 1) анализ запросов;             |
                                 | 2) чтение исходных данных       |
                                 |    для обработки из БД и их     |
     -------------     импорт    |    импорт;                      |
    | Файл        |<-------------| 3) запуск модуля обработки;     |
    | исходных    |     запуск   | 4) экспорт результатов обра-    |
    | данных      |       -------|    ботки и запись их в БД       |
     -------------       |        ---------------------------------
          |              |                       ^
          |              |                       |
          |              |                       |
          |              |                       |
          V              V                       |
      ------------------------------------------------------
      | Обрабатывающий     |------------------>| Файл        |
      | модуль             |                   | результатов |
      | QuickBASIC         |                   |             |
      ------------------------------------------------------
```

Рис. 1. Основная схема организации системы при обеспечении обмена информацией с базой данных средствами СУБД

Достоинством такой структуры системы является независимость обрабатывающих модулей от организации самой базы данных. Вместе с тем, она обладает следующими серьезными недостатками:

1. Так как обрабатывающие программы работают под управлением головного модуля, все они должны находиться в оперативной памяти одновременно. Соответственно под обрабатывающие программы может быть отведен весьма ограниченный объем памяти. Например, при работе с FoxBASE реальный объем оперативной памяти под модули обработки может составлять не более 100-150 Кбайт. Даже с учетом возможного разбиения подсистемы обработки на отдельные модули этого явно недостаточно (особенно, если иметь в виду достаточно высокие требования к оперативной памяти для программ графической обработки данных).

2. Перед запуском соответствующего обрабатывающего модуля весь объем необходимых исходных данных должен быть определен заранее. Получение дополнительных данных, потребность в которых возникла уже в ходе обработки, в данном варианте системы неосуществимо.

Возможен и другой вариант организации системы, при котором обрабатывающий модуль является головным. По мере поступления запросов на обработку данных он обращается к специальному модулю обмена с базой данных, реализованному в среде соответствующей СУБД, для ввода-вывода данных через файлы импорта-экспорта. Однако, наряду с присущими первому варианту недостатками, в данном случае еще и время обмена данными будет весьма значительным (требуется двойная загрузка модуля обмена данными — для импорта и экспорта, причем загрузка системы FoxBASE сама по себе занимает несколько секунд).

Принципиально другой подход к реализации системы открывает возможность автономного доступа к базе данных со стороны различных модулей системы (рис.2.). В этом случае достаточно просто решаются многие проблемы — и технические и организационные.

Во-первых, при этом практически снимаются ограничения на объем оперативной памяти для модулей системы, значительно сокращается время запуска отдельных операций и снижается нагрузка на диск (нет необходимости в создании временных файлов на диске).

Во-вторых, подсистемы накопления и обработки данных могут разрабатываться и эксплуатироваться совершенно независимо друг от друга. Таким образом, подсистема, реализованная нами на первом этапе работы, может быть оставлена в неизменном виде, а модули обработки — разрабатываться без оглядки на уже существующие структуры. Важным моментом практического использования системы является то, что накопление и обработка данных часто производятся совершенно различными службами заказчика, поэтому их совместное использование часто не только не обязательно, но и просто не нужно.

Для реализации такой структуры системы необходимо решить всего одну проблему — разработать подпрограммы обмена с файлами стандартных баз данных на соответствующем языке программироваввния, в нашем случае — на QuickBASIC. Мы так и сделали — создали библиотеку подпрограмм QB-DBF для работы с файлами баз данных типа .DBF и .DBT. Разработку подпрограмм обмена данными с использованием индексных файлов мы отложили на будущее, так как в этом пока нет особой необходимости (учитывая специфику обрабатывающих программ, для которых время выполнения операций поиска данных не является критичным).

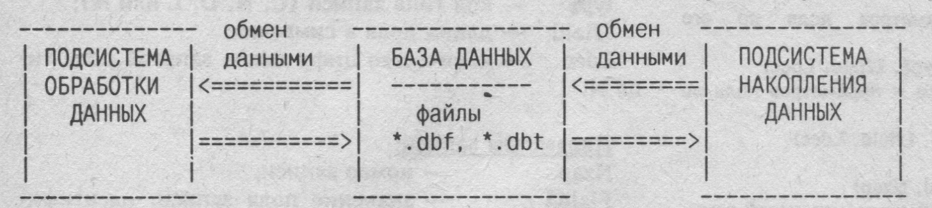


Рис.2. Схема организации системы при автономном обращении к базе данных различных модулей.

Несмотря на то, что библиотека разработана в рамках языка QuickBASIC, она может быть полезна и при работе с другими языками: документация содержит описание структуры файлов .DBF и .DBT, а тексты подпрограмм — алгоритмы, которые достаточно легко перевести на любой другой язык.

Общие сведения о библиотеке QB_DBF

Подпрограммы библиотеки QB_DBF выполняют следующие основные функции:

— поиск и открытие существующих файлов базы данных;

— получение информации о структуре базы данных;

— чтение, коррекция, создание, удаление записей;

— чтение, коррекция отдельного поля записи;

— преобразование полей типа N и M в вид, удобный для их последующей обработки.

В состав библиотеки QB_DBF входят подпрограммы модулей DBF_DBFR.BAS, DBF_DBFM.BAS, DBF_MEMO.BAS, DBF_SYMB.BAS, которые образуют 4 основные группы:

— обмена данными 1-го уровня (модули DBF_DBFM.BAS, DBF_MEMO.BAS);

— обмена данными 2-го уровня (модуль DBF_DBFR.BAS);

— преобразования полей типа N и M (модули DBF_SYMB.BAS, DBF_MEMO.BAS);

— вспомогательные подпрограммы (модуль DBF_SYMB.BAS).

Пользователь имеет возможность непосредственно работать с подпрограммами первых 3 групп.

При работе с подпрограммами обмена данными 1-го уровня все рабочие массивы и переменные резервируются в вызывающей программе пользователя. Зарезервировав и правильно используя необходимое количество блоков данных, пользователь может одновременно работать с несколькими базами данных и/или с несколькими записями. Такая организация может в некоторых случаях значительно сократить временные затраты, хотя и потребует незначительных раздумий при управлении данными.

Использование подпрограмм обмена данными 2-го уровня обеспечивает удобный доступ пользователя к одной базе данных. Работа с новой базой данных (операция открытия) может начинаться только после завершения работы с предыдущей (операция закрытия). Все рабочие массивы и переменные — описатели файла, содержимое записи и т.п. — резервируются в модуле DBF_DBFR.BAS, непосредственный доступ к которому со стороны пользователя невозможен.

Работа подпрограмм обмена данными 2-го уровня производится через подпрограммы 1-го уровня. Соответственно могут быть реализованы два варианта библиотеки:

— DBF_DBFR.LIB — полный состав библиотеки (можно использовать подпрограммы 1-го и 2-го уровней);

— DBF_DBFM.LIB — библиотека без подпрограмм 2-го уровня обмена данными (без модуля DBF_DBFR.BAS).

Состав подпрограмм библиотеки QB_DBF

Подпрограммы обмена данными 1-го уровня:

Модуль DBF_DBFM.BAS

1. Открытие базы данных
CALL DBF.Open (Ibuf(), Mfield$(), File$, Kzap, Kfield)
2. Закрытие базы данных
CALL DBF.Close (Ibuf())
3. Выборка информации о поле с именем NamF$
Nfield = DBF.Nfield (Mfield$(), NamF$, typ$, Lfield, Ldec)
4. Чтение описателя поля записи
CALL DBF.Field (Mfield$(), Nfield, NamF$, typ$, Lfield, Ldec)
5. Чтение записи с номером Nzap или обнуление содержимого записи
CALL DBF.GetZ (Ibuf(), Zsym$, Nzap)
6. Вывод на диск записи с номером Nzap
CALL DBF.PutZ (Ibuf(), Zsym$, Nzap)
7. Выборка поля записи
CALL DBF.Get.Field (Mfield$(), Field$, Zsym$, Nfield)
8. Коррекция (запись) нового значения поля в запись
CALL DBF.Put.Field (Mfield$(), Field$, Zsym$, Nfield)
9. Удаление записей файла
CALL DBF.Del (Ibuf(), Nzap, KzapNew)
10. Отмена признака "помеченная к удалению" для всех записей файла
CALL DBF.ReDel (Ibuf(), Kolz)

Модуль DBF_MEMO.BAS

1. Чтение текста поля из файла *.DBT
CALL DBF.Get.Memo(Ibuf(), Field$, Memo$, Nbl, Kbl)
2. Вывод текста поля в файл *.DBT
CALL DBF.Put.Memo (Ibuf(), Field$, Memo$, Nbl, Kbl)

Подпрограммы обмена данными 2-го уровня:

Модуль DBF_DBFR

1. Открытие базы данных
CALL DBFR.Open(File$, Kzap, Kfield)
2. Закрытие базы данных
CALL DBFR.Close
3. Определение имени и параметров поля по его порядковому номеру
CALL DBFR.Field(Nfield, NamF$, typ$, Lfield, Ldec)
4. Определение порядкового номера и параметров поля по его имени
Nfield = DBFR.Nfield(NamF$, typ$, Lfield, Ldec)
5. Чтение поля записи
CALL DBFR.Get.Field(Field$, Nfield, Nzap)
6. Чтение записи в память или формирование новой записи
CALL DBFR.GetZ(Nzap)
7. Вывод поля записи
CALL DBFR.Put.Field(Field$, Nfield, Nzap)
8. Сброс признака коррекции текущей информационной записи
CALL DBFR.ClearZ
9. Вывод текущей записи в базу с номером Nzap
CALL DBFR.PutZ(Nzap)
10. Удаление записи из файла
CALL DBFR.DelZ(Nzap, KzapNew)
11. Удаление "помеченных" записей
CALL DBFR.Del.All(KzapNew)
12. Очистка файла — удаление всех записей
CALL DBFR.Del.Clear(KzapNew)
13. Отмена признака "помеченная к удалению" для всех записей файла
CALL DBFR.ReDel (Kolz)

Подпрограммы преобразования полей типа N и M

Модуль DBF_MEMO (поле M)

1. Представление текста формата MEMO$ в виде массива MemoM$()
CALL DBF.Memo.GetM (Memo$, MemoM$(), kod)
2. Представление текстового массива MemoM$() в формат текста MEMO$
CALL DBF.Memo.PutM (Memo$, MemoM$(), KmemoM)

Модуль DBF_SYMB (поле N)

1. Преобразование поля типа N в число, Form$ — формат числа
Field$ = DBF.StrN$ (Anumb, Lfield, Ldec)
2. Преобразование вещественной переменной Anumb в символьное поле типа N
Anumb = DBF.Numb (Field$, Ldec, Form$)

Общепринятые обозначения формальных параметров

Параметры файла:

File$ — имя базы данных (имя файла *.DBF без расширения);

Kzap — количество записей в файле, полученное при открытии файла;

KzapNew — текущее значение количества записей в файле (полученное в результате коррекции длины файла);

Kfield — количество полей в записи;

Nfield — порядковый номер поля записи (= 0 — работа с байтом состояния записи).

Параметры поля записи:

NamF$ — имя поля;

typ$ — код типа записи (C, N, D, L или M);

Lfield — длина поля в символах;

Ldec — количество цифр после запятой поля типа N.

Параметры записи:

Nzap — номер записи;

Field$ — значение поля записи, идентифицируется порядковым номером Nfield, при этом значение Nfield = 0 соответствует работе с байтом состояния записи:

Field$ = "*" — "помеченная к удалению",
= " " — "непомеченная".

Модуль DBF_DBFM.BAS:

Ibuf(1 to 11) — блок управления базой;

Mfield$(1 to Kfield) — массив описателей полей определяется динамическим массивом в вызывающей программе, при отрытии базы его размерность переопределяется в подпрограмме DBF.Open;

Zsym$ — содержимое записи.

Модуль DBF_MEMO.BAS:

Memo$ — текст поля MEMO (содержимое записи файла *.dbt);

Nbl — номер начального блока: Nbl = VAL(Field$);

Kbl — количество блоков под данным текстом Memo$.

ПРАКТИЧЕСКАЯ РАБОТА С БИБЛИОТЕКОЙ QB_DBF

Схема основных действий при работе с библиотекой QB_DBF

1. Открытие базы данных — файлов .DBF и .DBT (файл .DBT открывается только, если есть поле MEMO).

2. Обмен данными (чтение/запись). Реализованы основные операции:

— определение параметров поля записи (дальнейшая обработка полей записи производится по порядковому номеру поля);

— чтение, коррекция (вывод в файл) отдельного поля записи, преобразование полей типа N и M в вид, удобный для их последующей обработки;

— работа с целой записью (чтение и вывод записи, создание новой записи, удаление отдельной записи или всех "помеченных" записей, отмена признака "помеченная" у всех записей файла и пр.).

3. Закрытие текущей базы данных:

— описатели базы освобождаются для работы с другой базой данных.

Организация работы с подпрограммами обмена 1-го уровня

При работе с подпрограммами обмена 1-го уровня резервирование и хранение описателей базы данных производится в вызывающей программе пользователя. Пользователь использует соответствующие массивы и переменные в течение всего сеанса работы с базой данных — от операции открытия до ее закрытия.

При работе с несколькими базами данных одновременно пользователь должен зарезервировать у себя в программе соответствующее количество служебных массивов для каждой базы.

Организация работы с подпрограммами обмена 2-го уровня

При открытии базы данных в модуле DBF_DBFR.BAS формируются блоки управления файлами и описателей полей. Обмен данными ФАЙЛ-ПАМЯТЬ выполняется на уровне отдельной записи. Сама запись хранится в модуле DBF_DBFR.BAS, в котором также фиксируется номер текущей (в памяти) записи NzapR и код ее состояния IcorrZ (= 0 — запись не корректировалась, т.е. ее содержимое в памяти и в файле идентично, = 1 — запись корректировалась, требуется ее вывод на диск). Обмен данными ПОЛЬЗОВАТЕЛЬ-ФАЙЛ выполняется на уровне отдельных полей записи.

Управление обменом ФАЙЛ-ПАМЯТЬ выполняется автоматически по такому алгоритму:

— если записи с заданным номером в памяти нет, то она читается с диска, но перед этим производится проверка состояния текущей записи;

— если запись корректировалась, то производится ее автоматический вывод на диск.

Таким образом, коррекция данных может осуществляться без выполнения операции вывода записи в файл пользователем в явном виде — вывод откорректированной или вновь созданной записи выполняется автоматически при чтении новой записи или закрытии базы.

Примеры применения подпрограмм библиотеки

Примеры реализации отдельных операций с файлами баз данных типа .DBF и .DBT с использованием подпрограмм библиотеки QB_DBF приведены в тексте модуля DBF_EXM.BAS и оформлены в виде отдельных процедур.

Представленные процедуры соответствуют варианту использования подпрограмм обмена данными 1-го уровня. Конструкции с использованием подпрограмм 2-го уровня показаны в тексте в виде комментария.

Программы модулей DBFM_D1.BAS и DBFR_D1.BAS демонстрируют вариант полного сеанса работы с базой данных через подпрограммы библиотеки.

Исходные тексты модулей DBF_EXM.BAS, DBFM_D1.BAS и DBFR_D1.BAS приводятся в конце статьи.

Варианты будущего развития библиотеки QB_DBF

Имеются два основных варианта развития функциональных возможностей библиотеки подпрограмм QB_DBF.

1. Разработка подпрограмм, выполняющих более сложные логические операции (например, поиск записи по ключу) через имеющийся набор подпрограмм 1-го или 2-го уровня. Очевидно, что эта задача по силам практически любому программисту средней квалификации.

2. Разработка подпрограмм, выполняющих более сложные логические операции, в рамках 1-го уровня подпрограмм, т.е. реализация специальных алгоритмов обмена ПАМЯТЬ-ДИСК, учитывающих особенности данной операции. Это может значительно оптимизировать время ее выполнения. Примером такого действия может являться операция поиска записи по ключу. Существующий набор подпрограмм ориентирован на чтение записи целиком. Создав подпрограмму чтения и сравнения только одного поля, вы значительно сократите время поиска для файлов с большой длиной записи. Примером такой подпрограммы является DBF.REDEL (модуль DBF_DBFM.BAS). Разумеется, для этого необходимо иметь некоторый опыт работы с файлами типа "BYNARY" и знать структуру файлов .DBF и .DBT.

Поставка программного продукта осуществляется в следующих вариантах:

— библиотека подпрограмм QB_DBF.LIB (объектные модули), описание состава и спецификаций подпрограмм;

— исходные тексты подпрограмм, описание структуры файлов .DBF и .DBT.

В состав этих двух вариантов поставки входят общее описание библиотеки, примеры реализации основных программных конструкций с использованием подпрограмм 1-го и 2-го уровней.

Второй вариант поставки будет полезен при создании аналогичных подпрограмм для других языков, обеспечит возможность эффективного расширения функциональных возможностей данной библиотеки.

Модуль DBF_EXM.BAS

```
DECLARE SUB DBF.Open (Ibuf%(), Mfield$(), File$,
                      Kzap%, Kfield%)
DECLARE SUB DBF.Del (Ibuf%(), Nzap%, Kz%)
DECLARE SUB DBF.Get.Memo (Ibuf%(), Field$,
                      Memo$, Nbl%, Kbl%)
DECLARE SUB DBF.Memo.GetM (Memo$, MemoM$(),
                      kod%)
DECLARE SUB DBF.Put.Memo (Ibuf%(), Field$,
                      Memo$, Nbl%, Kbl%)
DECLARE SUB DBF.Memo.PutM (Memo$, MemoM$(),
                      KmemoM%)
DECLARE SUB DBF.PutZ (Ibuf%(), Zsym$, Nzap%)
DECLARE SUB DBF.Put.Field (Mfield$(), Zsym$,
                      Field$, Nfield%)
DECLARE SUB DBFR.Put.Field (Field$, Nfield%,
                      Nzap%)
DECLARE SUB DBF.Get.Field (Mfield$(), Zsym$,
                      Field$, Nfield%)
DECLARE SUB DBF.GetZ (Ibuf%(), Zsym$, Nzap%)
DECLARE SUB DBF.Field (Mfield$(), Nfield%,
                NamF$, typ$, Lfield%, Ldec%)
DECLARE SUB DBFR.Field (Nfield%, NamF$, typ$,
                      Lfield%, Ldec%)
DECLARE SUB DBFR.Get.Field (Field$, Nfield%,
                      Nzap%)
DECLARE SUB DBFR.Open (File$, Kzap%, Kfield%)
DEFINT I-N
DECLARE FUNCTION DBF.Nfield (Mfield$(), NamF$,
                      typ$, Lfield, Ldec)
DECLARE FUNCTION DBFR.Nfield (NamF$, typ$,
                      Lfield, Ldec)
DECLARE FUNCTION DBF.StrN$ (Anumb, Lfield, Ldec)
DECLARE FUNCTION DBF.Numb (Field$, Ldec, Form$)
'*********************************************
'    Примеры работы с файлами типа *.DBF и *.DBT
'    с использованием подпрограмм библиотеки
'    QB_DBF.LIB:
'      1) обмена данными 1-го уровня (DBF_DBFM);
'      2) обмена данными 2-го уровня (DBF_DBFR).
'---------------------------------------------
'      Примеры представлены в виде текстов
'      подпрограмм
'  Подпрограммы относятся к  варианту DBF_DBFM,
'  соответствующие конструкции варианта DBF_DBFR
'      приведены в виде фрагментов комментариев
'*********************************************
'*** ОТКРЫТИЕ СУЩЕСТВУЮЩЕЙ БАЗЫ ДАННЫХ ***
      File$ = "Base"       ' имя базы данных без
                           ' расширения
                           ' открываются файлы
                           ' Base.dbf и
                           ' Base.dbt (если есть
                           ' поле MEMO)
 ----------------------------- DBFM -------------
DIM Ibuf(1 TO 15) 'резервируем массив под блок
                  'управления базой
      'фиктивное резервирование массива
      'описателей полей:
    REDIM Mfield$(1)
      'открываем базу данных, имя файла - File$
    CALL DBF.Open(Ibuf(), Mfield$(), File$,
                  Kzap, Kfield)
'
'   Если все OK (Kzap >= 0) - в подпрограмме
'                              DBF.open
'   производится переопределение массива:
'   REDIM Mfield$(1 TO Kfield)
'   и формирование массивов Ibuf() и Mfield$()
'
'   До момента завершения работы с базой данных
'   (закрытия) пользователь должен обеспечить
'   сохранность
'   массивов Ibuf() и Mfiled$().
'   Эти массивы используются в дальнейшем при
'   работе с подпрограммами обмена данными для
'   идентификации нужной базы данных
'-------------------- DBFR --------------------
'  CALL DBFR.Open(File$, Kzap, Kfield)
'=============================================
    ' Kzap - кол-во записей в файле
    ' Kfield - кол-во полей в записи
    ' анализ кода завершения операции открытия:
   IF Kzap = -1 THEN STOP    ' файла нет
   IF Kzap = -2 THEN STOP    ' Kzap > 32767 !
   IF Kzap = -3 THEN STOP    ' не найден файл
                         *.dbt(а был нужен!!)
END

DEFSNG I-N
DEFINT I-N
SUB DBFME.Copy (Ibuf(), Nzap.get, Nzap.put,
           Kzap)
'SUB DBFRE.Copy (Nzap.get, Nzap.put, Kzap)
'********** КОПИРОВАНИЕ ЗАПИСИ ФАЙЛА *******
'ВХОД:
'    Nzap.get - номер записи, ОТКУДА копируем
'             = 0 - очистка исходной записи
'    Nzap.put - номер записи, КУДА копируем
'             = -1 - добавить новую запись
'
'ВЫХОД:
'       Kzap - новое кол-во записей в файле,
'              если Nzap.put=-1
'              (требуется анализ: Kzap < 0 -
'              код ошибки)
'*********************************************
     Kzap = Nzap.put     ' при создании новой
                         ' записи Kzap
                         ' будет изменен
'-------------------------- DBFM -------------
' обнуление
      CALL DBF.GetZ(Ibuf(), Zsym$, Nzap.get)
' вывод новой записи
CALL DBF.PutZ(Ibuf(), Zsym$, Kzap)
'----------------------- DBFR -------------
```

```
     'CALL DBFR.GetZ(Nzap.get)
     'CALL DBFR.PutZ(Kzap)
'===============================================
END SUB

SUB DBFME.CORR (Ibuf(), Mfield$(), Kzap)
'SUB DBFRE.CORR (Kzap)
'** ПОИСК ЗАПИСЕЙ с ЗАДАННЫМ ЗНАЧЕНИЕМ ПОЛЯ ****
'** и КОРРЕКЦИЯ ДАННОГО ПОЛЯ НА НОВОЕ ЗНАЧЕНИЕ *
'  Kzap - кол-во записей в файле
'***********************************************
INPUT "Имя поля - "; NamF$
'
'    получение информации о поле NamF$
'--------------------------- DBFM --------------
Nfield = DBF.Nfield(Mfield$(), NamF$, typ$,
Lfield, Ldec)
'--------------------------- DBFR --------------
'Nfiled = DBFR.Nfield(NamF$, typ$, Lfield, Ldec)
'===============================================
'Nfield - пор. номер поля записи:
'дальше работаем с ним!!!!
'
     IF Nfield = 0 THEN PRINT " Поля с таким
              именем нет !!": EXIT SUB
'
'    ввод значения поля (считаем, что тип поля -
'    N или C)
     IF typ$ = "N" THEN
       INPUT "Искомое значение числового поля";
            Decal
       INPUT "Новое значение числового поля";
            DecalN
        ' преобразование десятичного значения в
        ' поле типа N
       FieldNew$ = DBF.StrN$(DeclaN, Lfield,
                           Ldec)
     ELSE
       INPUT "Искомое значение символьного
            поля"; Symb$
       INPUT "Новое значение символьного поля";
            FieldNew$
     END IF
'
'   поиск записи с заданным значением поля и ее
'   коррекция
    Kolz = 0     ' кол-во найденных и
                 ' откорректированных записей
    FOR Nzap = 1 TO Kzap
'--------------------------- DBFM --------------
CALL DBF.GetZ(Ibuf(), Zsym$, Nzap) ' чтение
                          ' записи - Zsym$
      CALL DBF.Get.Field(Mfield$(), Zsym$,
                    Field$, Nfield)
'--------------------------- DBFR --------------
'CALL DBFR.Get.Field(Field$, Nfield, Nzap)
'===============================================
IF typ$ = "N" THEN
      ' преобразование символьного поля в число
        Decalz = DBF Numb(Field$, Ldec, Form$)
        IF Decalz = Decal THEN GOSUB Out.File
   ELSE
        IF Symb$ = Field$ THEN GOSUB Out.File
   END IF
  NEXT Nzap
'     окончание просмотра файла
  PRINT "Кол-во откорректированных записей = ",
        Kolz
  EXIT SUB
'
'  вывод нового значения поля
Out.File:
  Kolz = Kolz + 1     ' счетчик откорректиро-
                      ' ванных записей
'--------------------------- DBFM --------------
CALL DBF.Put.Field(Mfield$(), Zsym$, Field$,
              Nfield)
CALL DBF.PutZ(Ibuf(), Zsym$, Nzap)   ' вывод
                             ' записи в файл
'--------------------------- DBFR --------------
' 1-й вариант - номер записи задан в явном виде:
     'CALL DBFR.Put.Field(Field$, Nfield, Nzap)
   ' 2-й вариант - работа с текущей записью:
     'CALL DBFR.Put.Field(Field$, Nfield, 0)
'===============================================
RETURN
END SUB

SUB DBFME.Creat.New (Ibuf(), Kzap)
'SUB DBFRE.Creat.New (Kzap)
'***** Добавить новую запись в конец файла *****
'ВЫХОД:
'        Kzap - новое кол-во записей в файле
'            (требуется анализ ошибки - Kzap < 0)
'***********************************************
Kzap = -1          ' код создания новой записи
'--------------------------- DBFM --------------
CALL DBF.GetZ(Ibuf(), Zsym$, 0)     ' обнуление
CALL DBF.PutZ(Ibuf(), Zsym$, Kzap) ' вывод новой
                                   ' записи
'--------------------------- DBFR --------------
'CALL DBFR.GetZ(0)
'CALL DBFR.PutZ(Kzap)
'===============================================
END SUB

SUB DBFME.DEL (Ibuf(), Nzap, Kzap)
'SUB DBFRE.DEL (Nzap, Kzap)
'***********************************************
'****************** УДАЛЕНИЕ ЗАПИСИ из файла
'        Nzap - номер удаляемой записи
'ВЫХОД:
'        Kzap - новое кол-во записей в файле
```

```
'           (требуется анализ: Kzap < 0 -
'           код ошибки)
'
'-------------------------- DBFM --------------
     CALL DBF.Del(Ibuf(), Nzap, Kzap)
'-------------------------- DBFR --------------
'CALL DBFR.DelZ(Nzap, Kzap)
'==============================================
END SUB

SUB DBFME.Memo.New (Ibuf(), Mfield$(), Nzap,
                    NamF$)
'SUB DBFRE.Memo.New (Nzap, NamF$)
'**********************************************
'**** ВВОД НОВОГО ЗНАЧЕНИЯ поля типа MEMO ******
'    Nzap - номер корректируемого поля
'    NamF$ - имя поля с типом MEMO
'ВНИМАНИЕ! Даже при записи НОВОГО значения поля
'MEMO следует предварительно считать в память
'его старое значение, так как только эта опера-
'ция позволит оптимизировать процедуру физичес-
'кого размещения данных в файле  dbt.
'
'-------------------------- DBFM --------------
     ' получение информации о поле NamF$:
   Nfield = DBF.Nfield(Mfield$(), NamF$, typ$,
                      Lfield, Ldec)
     ' чтение поля MEMO
   CALL DBF.GetZ(Ibuf(), Zsym$, Nzap)  ' чтение
                                       ' записи
   CALL DBF.Get.Field(Mfield$(), Zsym$, Field$,
             Nfield) ' чтение адреса поля
   CALL DBF.Get.Memo(Ibuf(), Field$, Memo$, Nbl,
                    Kbl)    ' чтение поля
     ' Значения Nbl и Kbl необходимо сохранить !
     ' Если  чтение поля MEMO не производилось,
     ' надо установить Nbl=0, Kbl=0
'-------------------------- DBFR --------------
'Nfiled = DBFR.Nfield(NamF$, typ$, Lfield, Ldec)
   'CALL DBFR.Get.Field(Field$, Nfield, Nzap)
'==============================================
DIM MemoM$(10) ' резервируем массив под
               ' текст поля
    PRINT "Вводите текст поля MEMO, пустая
          строка - конец ввода"
    Km = 0' счетчик строк
    DO
       LINE INPUT a$: IF a$ = "" THEN EXIT DO
       Km = Km + 1: MemoM$(Km) = a$
    LOOP
       ' преобразование символьного массива в
       ' формат строки MEMO в файле
    CALL DBF.Memo.PutM(Memo$, MemoM$(), Km)
'
'   запись в файл
'-------------------------- DBFM --------------
      CALL DBF.Put.Memo(Ibuf(), Field$, Memo$,
                       Nbl, Kbl)
      CALL DBF.Put.Field(Mfield$(), Zsym$,
                         Field$, Nfield)
      CALL DBF.PutZ(Ibuf(), Zsym$, Nzap)
' вывод записи в файл
'-------------------------- DBFR --------------
'CALL DBFR.Put.Field(Field$, Nfield, Nzap)
'==============================================
END SUB

SUB DBFME.MEMO.Out (Ibuf(), Mfield$())
'SUB DBFRE.MEMO.Out
'**********************************************
'* ВЫВОД на экран содержимого всех полей MEMO **
'********* одной записи файла *****************
   INPUT "Номер записи - "; Nzap
   ' Kfield - кол-во полей записи - определяется
   ' при отрытии файла
'-------------------------- DBFM --------------
   CALL DBF.GetZ(Ibuf(), Zsym$, Nzap)  ' чтение
                                  ' записи - Zsym$
'-------------------------- DBFR --------------
' чтение записи в явном виде не надо производить
'==============================================
FOR Nfield = 1 TO Kfield
       определение типа поля
'-------------------------- DBFM --------------
CALL DBF Field(Mfield$(), Nfield, NamF$, typ$,
               Lfield, Ldec)
'-------------------------- DBFR --------------
'CALL DBFR.Field(Nfield, NamF$, typ$, Lfield,
'               Ldec)
'==============================================
        IF typ$ = "M" THEN
           PRINT "Поле типа MEMO, имя поля - ";
                 NamF$
 -------------------------- DBFM --------------
CALL DBF.Get.Field(Mfield$(), Zsym$, Field$,
                   Nfield)
        CALL DBF.Get.Memo(Ibuf(), Field$,
                          Memo$, Nbl, Kbl)
'-------------------------- DBFR -------------
'CALL DBFR.Get.Field(Memo$, Nfield, Nzap)
'==============================================
' Memo$ - значение поля MEMO
           IF Memo$ = "" THEN
              PRINT " Пустое значение"
              ' резервируем массив для выделения
              ' полей текста MEMO$
              REDIM MemoM$(1 TO 1)
              CALL DBF.Memo.GetM(Memo$,
                                  MemoM$(), 0)
              FOR i = 1 TO UBOUND(MemoM$)
                 PRINT MemoM$(i)
              NEXT
           END IF
        END IF
```

```
   NEXT Nfield
END SUB

SUB DBFME.Met (Ibuf(), Mfield$(), Nzap, kod)
'SUB DBFRE.Met (Ibuf(), Mfield$(), Nzap, kod)
'***********************************************
'  kod = 0  - пометить запись как "удаленную"
'       > 0  - отменить ...
'    Nzap - номер удаляемой записи
'
     IF kod = 0 THEN Field$ = "*" ELSE Field$ =
" "
'--------------------------- DBFM --------------
   CALL DBF.GetZ(Ibuf(), Zsym$, Nzap) ' чтение
                              ' записи - Zsym$
   CALL DBF.Put.Field(Mfield$(), Zsym$, Field$,
                   0)    ' Nfield=0 !
   CALL DBF.PutZ(Ibuf(), Zsym$, Nzap)   ' вывод
                              ' записи в файл
'--------------------------- DBFR --------------
'CALL DBFR.Put.Field(Field$, 0, Nzap)
'===============================================
END SUB

SUB DBFME.STRUC.GET (Mfield$(), Kfield)
'SUB DBFRE.STRUC.GET (Kfield)
'***********************************************
'**** ЧТЕНИЕ СТРУКТУРЫ ЗАПИСИ БАЗЫ ДАННЫХ ******
'  Kfield - кол-во полей записи
''''''''''''''''''''''''''''''''''''''''
   PRINT " Описание полей:"
   PRINT "Имя ", "Тип", "Длина", "Дробь"
   FOR Nfield = 1 TO Kfield
'--------------------------- DBFM --------------
      CALL DBF.Field(Mfield$(), Nfield, NamF$,
                  typ$, Lfield, Ldec)
'--------------------------- DBFR --------------
'CALL DBFR.Field(Nfield, NamF$, typ$, Lfield,
            Ldec)
'===============================================
PRINT NamF$, typ$, Lfield, Ldec
   NEXT Nfield
END SUB
```

Модуль DBFM_D1.BAS

```
DECLARE SUB DBF.Close (Ibuf%())
DECLARE SUB DBFME.STRUC.GET (Mfield$(), Kfield%)
DECLARE SUB DBF.Get.Field (Mfield$(), Zsym$,
                        Field$, Nfield%)
DECLARE SUB DBF.Open (Ibuf%(), Mfield$(), File$,
                Kzap%, Kfield%)
DECLARE SUB DBF.Memo.GetM (Memo$, MemoM$(),
                        kod%)
DECLARE SUB DBF.Get.Memo (Ibuf%(), Field$,
                       Memo$, Nbl%, Kbl%)
DECLARE SUB DBF.GetZ (Ibuf%(), Zsym$, Nzap%)
DEFINT I-N
DECLARE FUNCTION IntSym (a$, Nsym)
DECLARE FUNCTION DBF.Nfield (Mfield$(), NamF$,
                             typ$, Lfield, Ldec)
DECLARE FUNCTION DBF.Numb (Field$, Ldec, Form$)
'
'    обращение к подпрограммам библиотеки
DBF_DBFM.LIB
'-----------------------------------------------
INPUT "Введите имя базы (без расширения) - ";
    File$
'-----------------------------------------------
     DIM Ibuf(1 TO 15)
' фиктивное резервирование массива описателей
' полей:
       REDIM Mfield$(1)
'   открытие файла + чтение реквизитов файла +
'   + перерезервирование массива описателей
'   полей и его чтение из файла
       CALL DBF.Open(Ibuf(), Mfield$(), File$,
                    Kzap, Kfield)
       IF Kz = 1 THEN STOP      ' файла нет
       IF Kz = 2 THEN STOP     ' это не *.dbf
'-----------------------------------------------
PRINT "Кол-во записей  = "; Kzap
    PRINT "Полей в записи  = "; Kfield
'
'    вывод на экран
    CALL DBFME.STRUC.GET(Mfield$(), Kfield)
'-----------------------------------------------
'   чтение записи с номером Nzap в символьную
'   константу Zsym$
   DO
     INPUT "Номер записи - "; Nzap
     IF Nzap = 0 THEN     ' конец работы
       CALL DBF.Close(Ibuf()): END
     END IF
     IF Nzap <= Kzap THEN EXIT DO    ' проверка
                               ' значимости номера
     PRINT " Ошибка в номере записи ";
   LOOP
'
'    чтение записи
    CALL DBF.GetZ(Ibuf(), Zsym$, Nzap)
'-----------------------------------------------
'    выборка поля
   DO
    INPUT "Имя поля - "; NamF$
'
'    получение информации о поле NamF$
    Nfield = DBF.Nfield(Mfield$(), NamF$, typ$,
                       Lfield, Ldec)
'   Nfield - пор. номер поля записи: дальше ра
'   ботаем с ним!!!!
'
    IF Nfield <> 0 THEN EXIT DO
    PRINT "Такого поля нет!";
   LOOP
   CALL DBF.Get.Field(Mfield$(), Zsym$,
                     Field$, Nfield)
```

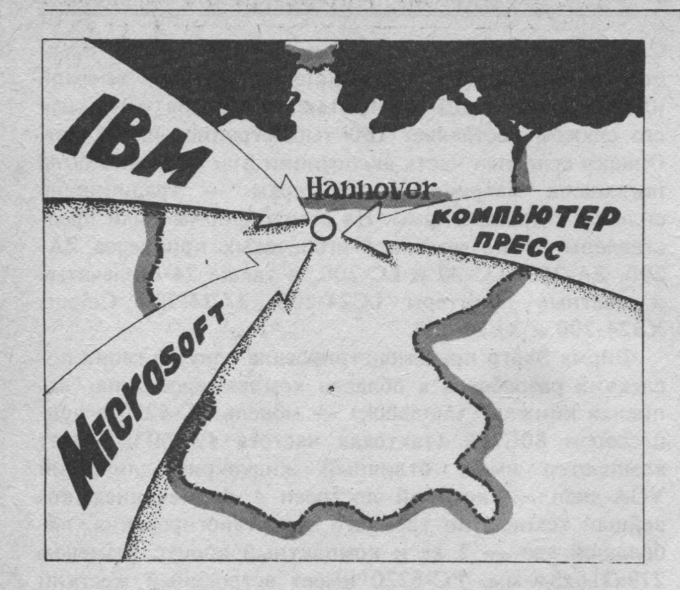
```
    PRINT "Поле - "; Field$
    IF typ$ = "N" THEN
      ' преобразование числового поля
      Decal = DBF.Numb(Field$, Ldec, Form$)
      PRINT "Decal, Form$ = "; : PRINT USING
                                 Form$; Decal
     END IF

    IF typ$ = "M" THEN
    ' чтение + обработка текста MEMO
      CALL DBF.Get.Memo(Ibuf(), Field$, Memo$,
                        Nbl, Kbl)
      IF Memo$ <> "" THEN
        ' резервируем массив для выделения
        ' полей текста MEMO$
        REDIM MemoM$(1 TO 1)
        CALL DBF.Memo.GetM(Memo$, MemoM$(), 0)
        FOR i = 1 TO UBOUND(MemoM$)
          PRINT MemoM$(i)
        NEXT
      END IF
    END IF
END
```

Модуль DBFR_D1.BAS

```
DECLARE SUB DBFR.close ()
DECLARE SUB DBFR.Field (Nfield%, NamF$, typ$,
                        Lfield%, Ldec%)
DECLARE SUB DBFR.Get.Field (Field$, Nfield%,
                            Nzap%)
DECLARE SUB DBFR.Open (File$, Kzap%, Kfield%)
DECLARE SUB DBF.Memo.GetM (Memo$, MemoM$(),
                           kod%)
DEFINT I-N
DECLARE FUNCTION DBFR.Nfield (NamF$, typ$,
Lfield, Ldec)
DECLARE FUNCTION DBF.Numb (Field$, Ldec, Form$)
'
'    обращение к подпрограммам библиотеки
DBF_DBFR.LIB
'----------------------------------------------
CLS
     INPUT "Введите имя базы (без расширения) -
"; File$
'----------------------------------------------
'  открытие базы данных
      CALL DBFR.Open(File$, Kzap, Kfield)
      IF Kz = 1 THEN STOP      ' файла нет
      IF Kz = 2 THEN STOP     ' это не *.dbf
'----------------------------------------------
    PRINT "Кол-во записей  = "; Kzap
    PRINT "Полей в записи  = "; Kfield
'
'    вывод на экран
    PRINT " Описание полей:"
    PRINT "Имя ", "Тип", "Длина", "Дробь"
    FOR Nfield = 1 TO Kfield
      CALL DBFR.Field(Nfield, NamF$, typ$,
                      Lfield, Ldec)
      PRINT NamF$, typ$, Lfield, Ldec
    NEXT Nfield
'----------------------------------------------
'   чтение записи с номером Nzap в символьную
'   константу Zsym$
  DO
    INPUT "Номер записи - "; Nzap
    IF Nzap = 0 THEN     ' конец работы
      CALL DBFR.close: END
    END IF
    IF Nzap <= Kzap THEN EXIT DO    ' проверка
                              ' значимости номера
    PRINT " Ошибка в номере записи ";
  LOOP
'
'   выборка поля
  DO
    INPUT "Имя поля - "; NamF$
'
'   получение информации о поле NamF$
    Nfield = DBFR.Nfield(NamF$, typ$, Lfield,
Ldec)
'   Nfield - пор. номер поля записи: дальше ра
'   ботаем с ним!!!!
'
    IF Nfield <> 0 THEN EXIT DO
    PRINT "Такого поля нет!";
  LOOP
'
    CALL DBFR.Get.Field(Field$, Nfield, Nzap)
'
    IF typ$ <> "M" THEN
      PRINT "Поле - "; Field$
'
      IF typ$ = "N" THEN
        ' преобразование числового поля
        Decal = DBF.Numb(Field$, Ldec, Form$)
        PRINT "Decal, Form$ = "; : PRINT USING
Form$; Decal
      END IF
    ELSE
    ' чтение + обработка текста MEMO
      Memo$ = Field$
      IF Memo$ <> "" THEN
        ' резервируем массив для выделения
        ' полей текста MEMO$
        REDIM MemoM$(1 TO 1)
        CALL DBF.Memo.GetM(Memo$, MemoM$(), 0)
        FOR i = 1 TO UBOUND(MemoM$)
          PRINT MemoM$(i)
        NEXT
      END IF
    END IF
END
```

По всем вопросам обращаться:
Москва, т.366-25-02, А.Колесов



Каджый, кто хотя бы в малой степени связан с миром компьютеров, знает, что изменения в этой области человеческой деятельности происходят с поистине космической скоростью. Прекрасную возможность проследить за этими изменениями, быть на острие передовых компьютерных технологий, охватить разом новинки сотен и тысяч фирм предоставляют международные выставки, многие из которых являются традиционными. Выставка CeBIT в ФРГ — одна из них. Корреспондент КомпьютерПресс был свидетелем этого значительного события, и сегодня — рассказ о CeBIT'91.

CeBIT'91

Ежегодно сотни тысяч посетителей устремляются на крупнейшую в мире компьютерную выставку CeBIT, которая проходит весной в Ганновере (ФРГ). В этом году выставка проводилась с 13 по 20 марта. По размаху и количеству представленных на ней фирм она в пять раз превзошла своего ближайшего конкурента — осеннюю выставку Comdex/Fall в Лас-Вегасе (США).

В CeBIT'91 приняли участие 4568 фирм из 45 стран мира, которые представили полный спектр современного программного и аппаратного обеспечения. Вся выставка, разместившаяся в 17 павильонах общей площадью 390 тыс. кв. м, была организована по 11 основным разделам:

— информационные системы;
— телекоммуникации;
— безопасность и надежность;
— программное обеспечение и базы данных;
— периферийное оборудование;
— автоматизация учреждений;
— банковские системы;
— микрокомпьютеры;
— сети;
— автоматизированные системы;
— обучающие системы.

Многие ведущие фирмы представляли свои изделия сразу по нескольким направлениям. Например, фирма Hewlett-Packard демонстрировала свои мини- и микроЭВМ, периферийное оборудование, программное обеспечение и сетевые разработки, фирма Sharp — конторское оборудование, портативные компьютеры, лазерные принтеры и т.п.

В отличие от предыдущих выставок на CeBIT'91 достаточно широко были представлены советские фирмы. Здесь можно было увидеть стенды "ПараГрафа", "Интермикро", "Аквариуса (ASI)" и некоторых других совместных и государственных предприятий. Так, на стенде ASI демонстрировались новые разработки фирмы, такие, как портативный компьютер типа "notebook" на базе 386 процессора, рабочая станция для локальных сетей, не имеющая системного блока, персональный компьютер на базе 286 процессора со значительно уменьшенным и более привлекательным, по сравнению с моделью ASI-286/12, системным блоком. Особое внимание привлекла модель новой и, безусловно, перспективной серии компьютеров ASI Megaline для процессоров 386 и 486.

В работе выставки принял участие чемпион мира по шахматам Гарри Каспаров, который на стенде фирмы IBM провел несколько партий с новой версией известной шахматной программы Deep Thought — Deep Thought 2.

Особый интерес у посетителей вызвал "говорящий" компьютер фирмы Toshiba. С помощью голосового интерфейса и забавной рожицы на экране компьютера создавалась атмосфера общения с живым существом. Компьютер мог сколь угодно долго поддерживать разговор на английском языке, причем практически на

любую тему, отпускать достаточно тонкие шутки и, что поражало посетителей более всего, запоминать голос говорящего с ним человека. Стоило лишь однажды представиться ему, и компьютер даже через несколько дней мог отличить ваш голос от многих других голосов: после вашего приветствия следовало четкое "Hello Valery" (если вы представились как Valery). Особенно удивительна была скорость, с которой выполнялась генерация ответов машины, что говорит об исключительной эффективности алгоритмов обработки речи и нахождения адекватных путей ведения диалога.

Конечно же, рассказать обо всех событиях и экспонатах выставки CeBIT'91 невозможно, поэтому мы остановимся лишь на некоторых из них.

Фирма Hewlett-Packard демонстрировала свой новый лазерный принтер LaserJet IIISi. Этот принтер продолжает линию принтеров LaserJet III, что позволяет практически без дополнительного чтения руководства приступить к работе, если до этого вы были знакомы с LaserJet III или LaserJet IIID. Это обусловлено единообразием конструкции передней панели управления и командами, которые появляются на жидкокристаллическом экране.

Особенностью LaserJet IIISi, безусловно, является скорость его работы — 16 страниц в минуту при формате листа A4 и 17 страниц при формате Letter. Такая высокая скорость обеспечивается благодаря наличию в LaserJet IIISi форматера страниц на базе RISC-процессора. Принтер имеет два поддона для бумаги на 500 листов формата A4 каждый, а также дополнительное устройство для подачи конвертов (емкость 100 шт.).

Разрешающая способность принтера LaserJet IIISi по-прежнему равна 300 точкам на дюйм, однако за счет использования технологии улучшения разрешающей способности Resolution Enhancement, которая стала применяться фирмой Hewlett-Packard в принтерах серии III, а также использования специального мелкозернистого тонерного порошка, разрешение LaserJet IIISi сравнимо с качеством печати уровня 600 точек на дюйм.

В стандартной конфигурации принтер поставляется с памятью 2 Мбайта, однако наличие 4 разъемов для установки дополнительных блоков памяти позволяет довести ее общую емкость до значительной величины в 17 Мбайт.

По словам специалистов фирмы Hewlett-Packard, все эти характеристики, а также возможность установки дополнительных (опциональных) сетевых интерфейсов типа AppelTalk, Ethernet и Token Ring делают лазерный принтер LaserJet IIISi идеальным печатающим устройством для коллективного использования в среде локальных сетей.

Фирма Star Micronics также представила свой новый лазерный принтер LaserPrinter 4, предназначенный в основном для использования в качестве персонального печатающего устройства. Его разрешающая способность в 300 точек на дюйм, достаточно удобная панель управления и скорость работы — 4 страницы в минуту вполне соответствуют принтерам этого класса. Следует отметить, что, согласно утверждению представителей фирмы, LaserPrinter 4 обладает высокой надежностью и долговечностью, благодаря чему срок его службы составляет 150 тыс. страниц, или 5 лет. Однако основная часть экспозиции Star Micronics была посвящена матричным принтерам — традиционно сильной стороне фирмы. На стенде фирмы были представлены новые модели 9-игольчатых принтеров ZA-200, ZA-250, LC-20 и LC-200, а также 24-игольчатые и цветные принтеры LC24-200, LC24-200 Colour, XB24-200 и XB24-250.

Фирма Sharp продемонстрировала одну из своих последних разработок в области компьютеров типа "записная книжка" (notebook) — модель PC-6220 с процессором 80C286 (тактовая частота 12 МГц). Этот компьютер имеет отличный жидкокристаллический VGA-дисплей, который построен с применением новейшей технологии тройного супертвистирования, небольшой вес — 2 кг и компактный корпус размером 279x216x34 мм. PC-6220 имеет встроенный жесткий диск на 20 Мбайт с временем доступа 23 мс, оперативную память емкостью 1 Мбайт с возможностью расширения до 3 Мбайт и ПЗУ, которое содержит операционную систему MS-DOS 4.01 и известную программу Lap-Link для обмена данными между компьютерами. Компьютер обладает широкой гаммой дополнительных устройств, таких, как блок с цифровой клавиатурой, накопитель 3-дюймовых гибких дисков емкостью 1.44 Мбайта с портами для подключения 5-дюймового накопителя гибких дисков и расширенной клавиатуры стандарта IBM, а также блок расширения с адаптером VGA-дисплея, платой модем/факс и местами для установки 2 плат стандарта AT. Если учесть, что с дополнительным блоком для батарей PC-6220 может работать до 5 часов, становится понятным, что фирма Sharp приподнесла хороший подарок пользователям, нуждающимся в легком и в то же время достаточно мощном переносном компьютере.

На стенде фирмы Novell были представлены новые версии популярной сетевой операционной системы NetWare: NetWare 286 2.2 и NetWare 386 3.11. Достаточно впечатляюще выглядела демонстрация работы NetWare 386 3.11 для 250 рабочих станций — одного из новшеств этой версии по сравнению с предыдущей версией.

В заключение хотелось бы отметить, что благодаря стремительному развитию рынка восточноевропейских стран, в том числе Советского Союза, выставка CeBIT приобретает особое значение. Многие из их фирм устремляются именно на CeBIT в поиске выгодных контрактов. Это позволяет предположить, что в ближайшее время Ганновер станет именно тем местом, где фирмы Запада и Востока смогут способствовать столь необходимым экономическим реформам в восточноевропейских странах, которые невозможны без использования передовых компьютерных технологий.

В.Миропольский



Сегодня мы публикуем выступление Питера Нортона на открытии выставки Comtek' 91, состоявшемся в Москве 8 апреля 1991 года.

ВЗГЛЯД В ПРОШЛОЕ, ПЕРСПЕКТИВЫ БУДУЩЕГО

Введение

Удивлен сходством той ситуации, в которой вы сегодня находитесь, с тем, что мне пришлось самому испытать в начале семидесятых годов.

В Советском Союзе происходят большие перемены, и вместе с ними у людей возникает все больше возможностей начать свой бизнес. Надежды возлагаются на способных людей с их прекрасными идеями, а также на большое количество потенциальных покупателей с их неудовлетворенными потребностями.

И успех возможен как для малых предприятий с довольно скромными ресурсами, так и для большого бизнеса.

Советские предприниматели имеют сейчас те же заманчивые перспективы, что и я в свое время, или мои коллеги — Митч Кэйпор, который написал Lotus 1-2-3, или Стив Джобс, основавший Apple Computers, или Билл Гейтс, президент Microsoft — и сотни других людей, ставших первопроходцами американской индустрии микрокомпьютеров.

Когда я начал заниматься программированием в начале семидесятых, персональных компьютеров было очень мало. Появление IBM PC в 1982 году способствовало быстрому росту отрасли. Большая часть инфраструктуры, которую мы сейчас имеем в США с ее доступом к капиталу, каналами распространения, образованием, и так далее, начала формироваться именно в то время.

Мы знали, что нам нужно сделать, чтобы решить те сложные задачи, которые мы ставили перед собой, но вряд ли кто-нибудь представлял себе, к каким грандиозным результатам приведет развитие новой индустрии.

У вас уже есть то, чего не было у нас, то, что может быть очень ценным на этом этапе развития советской экономики. У вас накоплен колоссальный опыт в импровизации, в умении работать только с тем, что у вас есть под рукой, в способности полагаться на самих себя. У вас есть то, что называется "житейской мудростью". Знание, как извлечь выгоду из того, что есть, и выйти из любой ситуации — это отличный опыт, который можно с успехом использовать, чтобы открыть свое дело.

И поскольку история повторяется, я бы хотел поделиться с вами своим опытом.

Начало

Когда вы начинаете свое дело, вам надо быть готовым к тому, чтобы засучить рукава и делать все самостоятельно. Когда я написал свою первую программу "UnErase", практически никакой помощи со стороны не было, чтобы попытаться получить какую-то выгоду от существования этой программы. У меня не было ни опыта, ни образования в области бизнеса. Все, что у меня было — мои сбережения. Но я чувствовал, что сделал отличную программу, и мне не хотелось останавливаться. И я хотел делать все сам.

Заказы на первые программы в 1982 году я выполнял у себя дома, используя персональный компьютер для копирования программ на гибкие диски, а затем шел на почту и отправлял их своим заказчикам.

Создание программы: первый раунд

Создание хорошей программы — трудное дело. Существует два подхода, которые вы можете при этом

использовать. Оба могут быть успешны. Все зависит от обстоятельств.

Подход первый — это полагаться во всем на себя. Второй — полагаться на вашего заказчика.

Если вы верите себе, если интуиция подсказывает вам, что вы знаете, что может понадобиться людям — следуйте вашей интуиции и не слушайте, что вам говорят со всех сторон. Если интуиция вас не обманывает, тогда, положившись на себя, Вы непременно придете к успеху.

Второй подход — положиться на заказчика. Спросите ваших потенциальных пользователей, что им нужно. Вы узнаете много интересного, поговорив с каждым из них.

Преимущество первого подхода заключается в том, что у вас появляется хорошая интуиция, некое видение будущего — вы можете знать, что пользователям понадобится завтра, в то время, как сами пользователи могут этого не знать. Очень часто пользователей беспокоят проблемы вчерашнего дня, а главная задача для нас, бизнесменов — создавать программы, которые будут нужны завтра.

Например, в 1982 году, когда я написал свою первую программу "UnErase", чтобы восстановить случайно стертый файл, никому такие программы, казалось, были не нужны. Но я-то знал, какую ценность для меня и для моих друзей представляет написанная мною программа. Спустя некоторое время, когда все поняли, насколько важно иметь возможность восстановить утерянные данные, "UnErase" фактически сформировала новый сектор рынка программных средств для PC, который называется сектором сервисных программ.

Преимущество второго подхода в том, что вы узнаете, что представляет собой ваш потенциальный заказчик, каковы его потребности. Вы узнаете мнения многих людей. Такой подход мы называем "управляемый пользователь".

Оба этих подхода могут работать. Вам надо понять, какой из них больше подходит к определенной ситуации.

В день открытия выставки Comtek-91 прошла презентация шести русифицированных продуктов фирмы Symantec. На ней выступили П.Нортон — сейчас он входит в совет директоров фирмы Symantec и С.Тирелл — менеджер по Восточной Европе и СССР.

Сегодня Symantec является одной из крупнейших компаний США, число ее сотрудников приближается к 600. С момента создания фирма активно развивала международную торговлю. Ее программные продукты переведены на 11 языков.

В 1990 году произошло слияние фирм Peter Norton Computing и Symantec. Это позволило объединить колоссальный творческий потенциал программистов первой с богатым опытом производства и сбыта программных продуктов, накопленным второй из фирм.

Сейчас Symantec имеет вспомогательные фирмы в 7 странах мира, сотрудничая, кроме того, со всеми крупнейшими странами мира. Фирма активно работает над установлением деловых отношений с Россией и другими нашими республиками, странами Центральной Европы и рассматривает это как долгосрочную работу, направленную в будущее. По словам Питера Нортона, "надежды возлагаются на способных людей с их прекрасными идеями, а также на большое количество потенциальных покупателей с их неудовлетворенными потребностями".

По мнению фирмы, успех компьютеризации в нашей стране тесно связан с тем, как будет развиваться производство компьютеров и как будет расти количество фирм, работающих в области информатики. Представители Symantec высказали надежду на то, что мы добьемся успеха в этой области.

Создание программы: второй раунд

Совершенно очевидно, что создание одной программы не может обеспечить длительный успех. Параллельно с разработкой первого продукта необходимо думать о перспективе. Существует два способа двигаться дальше. Первый — улучшить существующий продукт, второй — создать новый.

Первая опция: усовершенствовать продукт. Создание новой версии существующего продукта — если он пользуется успехом — одна из наиболее важных конкурентоспособных характеристик, которую вы можете использовать в условиях свободного предпринимательства. Во-первых, пользователь уже имеет ваш продукт, и вам не потребуется прилагать так же много усилий для продажи новой версии, как для продажи предшествующей. Он вложил время и деньги в изучение программы и знаком с ней. Во-вторых, вы используете созданный продукт и добавляете к нему новые функции. Это гораздо быстрее и дешевле, чем начинать с нуля.

Мы не раз обновляли и усовершенствовали наши сервисные программы, Каждая новая версия предстает более мощной, качественной, функционально насыщенной. Фактически только в прошлом году мы выпустили Norton Utiliuties 5.0. Это пятая большая версия — вместе с другими программами — той самой первой "UnErase", которую я создал в 1982 году. А что касается Norton Commander — мы сделали уже третью версию.

Так что, как вы сами видите, усовершенствование продукта, который хорошо зарекомендовал себя, может быть весьма прибыльным делом.

Вторая опция: создать новый продукт. Вы можете создать продукт, которого вообще нет на рынке, или найти хороший продукт, который, несмотря на конкуренцию, не обновляется производителем. И лучшее, что вы можете сделать в этой ситуации — предложить лучшую версию конкурентного продукта, а не создавать полностью новый.

Мы, специалисты по высоким технологиям, зачастую сосредоточиваемся только на самых современных разработках, охотимся за наиболее совершенными продуктами и восхищаемся новыми технологиями. Но, как я узнал, Вашего пользователя просто так не вдохновишь.

Бизнесмену нужно знать, что большинство пользователей персональных компьютеров выполняют общие задачи и используют базовые продукты типа Lotus 1-2-3. У них нет ни времени, ни желания, чтобы перейти на использование современной технологии. Помните об этом при создании нового продукта.

Интеллектуальная собственность

Это еще одна тема, о которой в последнее время говорят все больше и больше. В США этот вопрос также весьма активно обсуждается — особенно применительно к программному обеспечению. Я знаю, что многие из вас имеют копии некоторых моих программ, и очень приятно видеть, что они так популярны здесь.

Когда я начинаю распространять продукт, я хочу, чтобы мои пользователи могли извлечь из него максимальную пользу. Но это означает, что им нужно руководство по эксплуатации, сопровождающее программу. Им надо попробовать поработать со всеми функциями и, может быть, если им понадобится дополнительная помощь, они обратятся в отдел поддержки пользователей, чтобы получить дополнительную подготовку. Всем нам, представителям компьютерной индустрии, выгодно поддерживать права интеллектуальной собственности, защитить свой бизнес.

Маркетинг и продажа: больше, чем половина задачи

Создание хороших программ — это только часть той сложной задачи, с которой мне пришлось столкнуться после того, как я написал "UnErase". Одна из любопытных особенностей свободной экономики в том, что маркетинг и продажа продукта — это более, чем ползадачи. У нас даже поговорка есть, что самый прекрасный программный продукт не сможет продать себя в условиях свободного предпринимательства.

Когда я предложил свою первую программу, я прилагал массу усилий, чтобы довести ее до пользователей. К тому времени я уже несколько лет работал с компьютерами, но только тогда начал использовать IBM PC, распространяемый на коммерческой основе. Как только я научился работать с компьютером, я решил поделиться своими навыками со всеми пользователями посредством распространения информационного листка. Я назвал этот листок "шпаргалкой".

Шпаргалки содержали много практической информации и имели, конечно же, рекламу моих собственных продуктов на обратной стороне каждого листка. Каждые два месяца я посылал мои шпаргалки по почте в компьютерные клубы и магазины — например, ComputerLand. Главной целью было сделать мои программы известными и внушить пользователям доверие к ним.

Важность маркетинга поистине трудно переоценить. Я использовал еще одно средство для популяризации моих продуктов. Я понял, что, в общем, технологическим продуктам недостает человеческого тепла. И я хотел, чтобы мой продукт имел эти человеческие черты, поэтому я поместил свое имя на него, назвал своим именем компанию, а на коробках начали печатать мои фотографии.

Пользователь как бы получает сообщение: "За этой программой стоит не безликая корпорация, а реальный человек, который знает, что делает".

Это всего лишь два примера использования мною средств для целей маркетинга. Эти средства так же зависят от конкретной ситуации и от ваших собственных возможностей, конечно. Оказалось, что я могу хорошо объяснить пользователю-новичку технические аспекты программы — вероятно, потому, что я хорошо помню свой собственный опыт на IBM PC. И писать те самые шпаргалки было для меня вполне естественным занятием.

Выступление Питера Нортона на презентации продуктов фирмы Symantec

Спасибо, я очень рад присутствовать на Comtek. Это очень интересное время — время моего приезда в Советский Союз. Это время больших перемен и больших возможностей.

Мы очень рады принять участие в развитии компьютерных технологий, в частности программного обеспечения в СССР.

Я очень рад сегодня встретится с вами, и горд тем, что мы представляем русские версии наших программных продуктов советским программистам.

Счастлив приветствовать вас на Comtek-91 и внести свой вклад в это новое начинание.

Управление бизнесом: просто технических знаний недостаточно

Я говорил немного о важности маркетинга и продаж, что подвело меня к другой теме: важность разумного управления бизнесом.

Выступление Санни Тирелл на презентации продуктов фирмы Symantec

Год назад фирма Symantec приняла участие в первой международной выставке Comtek-90. Тогда многие фирмы приехали, чтобы посмотреть, какова ситуация на советском рынке компьютерных технологий.

И сегодня, год спустя, мы очень горды, что можем представить шесть продуктов Symantec, переведенных на русский язык. Программы, которые мы выбрали для распространения в Советском Союзе, следующие: Norton Commander, Norton Utilites, Norton Backup, Norton AntiVirus, Time Line и Q&A. Мы хотели бы, чтобы эти продукты были у советского пользователя, поэтому они будут продаваться за рубли.

Это самая большая страна из всех, где мы когда-либо работали. И чтобы работа была успешной, мы установили отношения с тремя квалифицированными дистрибьюторами: СП "Компьютерные технологии", СП "Диалог" при МИФИ и ГИЦ "ЦЕНТРПРОГРАММСИСТЕМ".

Наши дистрибьюторы позволят нам иметь сеть дилеров, ориентированных на конечного пользователя. Поддержка пользователя, предоставляемая ими, будет включать постоянно действующую горячую линию, консультации и проведение семинаров. Кроме того, дистрибьюторы позволят надежно и качественно обслуживать пользователей и снабжать их последними версиями пакетов. Наконец, дистрибьюторы будут обучать пользователей работе со всеми видами наших продуктов.

На базе СП "Компьютерные технологии" создается клуб пользователей Питера Нортона, который позволит покупателям программ обмениваться информацией и идеями.

Кроме того, есть желание создать клуб пользователей пакета Time Line, но детали этого проекта пока неясны.

Часть доходов от распространения продукции Symantec в СССР пойдет на благотворительные цели. Десять рублей с каждой проданной копии программы будут направляться на благотворительный счет "Огонек-АнтиСПИД" для финансирования закупок одноразового медицинского оборудования.

Когда я основал Peter Norton Computing в 1982 году, у меня не было никакого коммерческого образования. Я занимался программированием для аэрокосмической промышленности в Южной Калифорнии. Я имел математическое образование, накопил приличные знания в информатике. Но у меня не было ни малейшего представления о том, как начать свой бизнес и какие надо прилагать усилия, чтобы он был успешным.

Мне приходилось одновременно изучать теорию и практику бизнеса, надо было знать все: от бухгалтерии и управления персоналом до маркетинга и продаж.

Свое образование в области управления я получал "на работе", действуя методом проб и ошибок, чтобы понять, что работает и что нет. Я читал все, что попадало мне в руки — Fortune, Forbes, Business week, и вообще все, что тогда было написано о бизнесе. И говорил, с кем только мог.

Мораль: учитесь управлять — учитесь где можете и как можете.

Оценка своей силы

Небольшие компании обычно основываются умными, талантливыми и опытными людьми. И мне очень приятно видеть их сегодня в этом зале.

Я бы хотел посоветовать вам реально оценить свои силы и найти людей, которые дополнили бы их опытом в других областях.

И когда вы найдете этих талантливых людей, предоставьте им возможность использовать этот опыт.

В некоторых случаях, развивая это направление, можно соединить опыт двух компаний. В середине прошлого года я объединил свою компанию с фирмой Symantic. Мы объединили опыт создания сервисных программ и хорошо действующую торговую сеть. Symantic выпускает и другие программные продукты — управление проектами, коммуникации, языки программирования.

Оценка возможностей компании не менее важна, чем оценка возможностей каждого отдельного человека.

Развитие технологий

Шесть лет назад я говорил в Болгарии и повторю сейчас: использование персональных компьютеров сулит большую выгоду малому и среднему предпринимательству, но предпринимателям не надо иметь обязательно 386 или 486 микропроцессоры, которые так нужны нам — специалистам в области технологий.

Бизнесмен может использовать доступную, простую в эксплуатации и сравнительно дешевую систему. Вы можете значительно улучшить свою производительность при минимальных затратах, приобретя компьютер на 286 микропроцессоре.

Заключение

В заключение, делая вывод из собственного опыта, я хотел бы сказать вам, что если вы действительно заинтересованы, готовы идти на риск, если вы изобретательны — вы добьетесь успеха.

Я высоко ценю Ваше умение импровизации, умение использовать жизненный опыт. Это очень важные качества для того, чтобы начать бизнес, чтобы стать преуспевающим предпринимателем.



8 апреля 1991 года в Москве на международной выставке "КОМТЕК-91" проходила пресс-конференция фсемирно известной компании Intel. Мы представляем вашему вниманию материалы пресс-конференции, а также тексты выступлений директора по стратегическим проектам фирмы Дмитрия Ротова и регионального менеджера Бенни Гинмана.

Intel В СССР

8 апреля 1991 года корпорация Intel подписала свои первые дистрибьюторские соглашения в СССР. Два совместных предприятия, уже достаточно хорошо зарекомендовавшие себя на советском рынке, "Компьютерные технологии" (КоТ) и "Диалог" будут выступать в качестве официальных дистрибьюторов продукции компании Intel. Соглашения вступают в силу с момента подписания.

"Эти компании обладают всем, что необходимо "Интелу", — сказал Бени Гинман, Региональный Менеджер по СССР. — "Они имеют достаточный опыт и развитую сеть, необходимую для удовлетворения растущих потребностей в автоматизации учрежденческой деятельности и запросов изготовителей комплексного оборудования".

"Наша политика в Советском Союзе ничем не отличается от той, которую мы проводим в других странах. В ее основе лежит комплексный подход, который предусматривает продажу нашего оборудования на разных уровнях интеграции. В настоящее время мы ведем переговоры с потенциальными дистрибьюторами с целью продажи наших компонентов непосредственно изготовителям комплексного оборудования", — добавил Гинман.

Компания также объявила о своем намерении открыть в этом году свой офис в Москве, где будут работать как американские, так и советские специалисты.

Корпорация Intel
и ее стратегия в Советском Союзе
Дмитрий Ротов

Дамы и господа,

Корпорация Intel была основана в калифорнийской "Кремниевой долине" в 1968 году и добилась успеха благодаря изобретению и производству целого ряда электронных компонентов, используемых сейчас в каждом компьютере. Среди изобретений компании Intel, например, динамическое оперативное запоминающее устройство (ДОЗУ), стираемое перепрограммируемое постоянное запоминающее устройство (СППЗУ), а также первый микропроцессор.

В настоящее время наша компания известна как производитель микропроцессоров 8086, 80286, Intel 386 (TM), а также Intel 486 (TM), которые поистине являются "сердцем" персональных компьютеров, совместимых с IBM. По последним подсчетам, 80 миллионов персональных компьютеров во всем мире используют микропроцессоры, созданные по технологии Intel. Эти микроЭВМ производятся тысячами компаний по всему миру: в США, Европе, Японии, Корее, Тайване, а также в Советском Союзе. Компьютеры компании Intel обрабатывают больше программ, чем какие-либо другие ЭВМ. Сегодня Intel сохраняет свое технологическое лидерство в области микроэлектроники и явля-

КомпьютерПресс: Налагает ли КОКОМ какие-нибудь ограничения на продажу продукции Intel в Советском Союзе?

Интел: Не такие строгие, как раньше. Уже сейчас снят запрет на все микрокомпьютерные платформы на базе i386/33 DX и программное обеспечение. Экспорт i486 пока что невозможен без лицензии.

КП: Будете ли вы продавать свои изделия за рубли?

И: По крайней мере в ближайшее время — только за валюту.

ется крупнейшим поставщиком компонентов для микроЭВМ, модулей и систем, таким образом предлагая изделия всех уровней интеграции. Традиционно компоненты составляют 70% нашего бизнеса.

Наша основная цель в Советском Союзе заключается в обеспечении доступа растущей советской электронной промышленности к нашим компонентам и изделиям системного уровня. Относительно компонентной стороны нашего бизнеса я должен сказать, что на нас произвели огромное впечатление возможности советской промышленности. За последние несколько месяцев мы были на многих советских электронных заводах, ознакомились с их техническими и деловыми возможностями, и нам стало совершенно ясно, что, имея доступ к микропроцессорам Intel и другим ключевым компонентам, они смогут наладить производство персональных компьютеров на своей территории.

Доступ к широкому ассортименту продукции Intel обеспечит рост советской компьютерной промышленности и позволит ей значительно увеличить производство микроЭВМ. Мы намерены содействовать этому процессу, организовав обучение советских специалистов для успешного использования наших компонентов. Нам также хотелось бы помочь другим отраслям советской промышленности в разработке и применении других изделий (таких как, например, микроконтроллеры) в производимом ими промышленном оборудовании. Во многих случаях использование одного единственного микроконтроллера стоимостью менее 5 долларов может сэкономить механические части ценой в несколько сот долларов. Это также обеспечит надежность, высокую производительность и позволит снизить цену на такое сложное промышленное оборудование, как, например фрезерный станок.

И наконец, наши 32-разрядные процессоры семейств 386 и 486 позволят советской электронной промышленности отказаться от уже устаревших на Западе 16-разрядных микроЭВМ и, таким образом, выйти на качественно новый уровень. Мы также предоставим Советскому Союзу наши 32-разрядные изделия в системной форме, что будет способствовать переходу к 32-разрядным настольным комьютерам.

Неотъемлемой частью стратегии компании Intel является поставка продукции всех уровней интеграции. Многим заказчикам на Западе необходима наша продукция (или платы, или изделия системного уровня) с тем, чтобы выйти на рынок быстрее, чем если бы они разрабатывали и производили ее сами. Наши последние достижения в области компьютерной технологии предоставляют программистам и пользователям возможность начать работать на самом современном оборудовании.

Первое дистрибьюторское соглашение Intel в СССР

Бени Гинман

Дамы и господа,

Для меня и моей компании большая честь присутствовать здесь сегодня и сообщить вам о первом значительном шаге корпорации Intel на пути организации торговой сети и системы обслуживания, предназначенных для потенциальных заказчиков в СССР. Для компании Intel выход на новый рынок исторически всегда означал поиск дистрибьютора. Поскольку 30-40% продаж компании осуществляется через официальных дистрибьюторов, Intel уделяет отношениям с ними

КомпьютерПресс: Расскажите вкратце о вашем новом кристалле i586 и о том, когда вы собираетесь приступить к выпуску этого процессора.

Интел: Вообще, мы не уполномочены давать комментарии по этому поводу, но можем сказать, что при степени интеграции около одного миллиона транзисторов, микропроцессор 80586 будет работать примерно в 4-5 раз быстрее своего предшественника.

КП: Будет ли фирма продолжать линию RISC-процессоров?

И: Обязательно, причем уже в этом году было объявлено о начале производства двух новых микропроцессоров, совместимых с i860.

КП: Не чувствуете ли вы жаркое дыхание догоняющих вас конкурентов, таких, как, например компания Motorola?

И: В сфере микропроцессоров — нет! Наша фирма является сегодня признанным лидером как с точки зрения концептуально новых разработок, так и по объемам производства. А к середине 90-х марка Intel будет стоять на четырех из каждых пяти выпускаемых во всем мире микропроцессоров.

первостепенное значение. Немало времени и сил уходит на формирование дистрибьюторской сети, так же как и на обеспечение необходимого уровня профессионализма персонала. Именно поэтому в каждой стране существует ограниченное число дистрибьюторов. И для компании Intel, и для ее дистрибьюторов установление взаимоотношений является задачей огромной важности, так как это означает формирование сети заказчиков. Это объясняет тот факт, что многие из наших европейских дистрибьюторов являются нашими партнерами с момента зарождения Intel. Компания Intel была образована в 1968 году, и уже в 1969 году был назначен наш первый европейский дистрибьютор в Швеции. Как вы понимаете, за это время мы хорошо узнали наших дистрибьюторов, как и они нас, нашу продукцию и нашу стратегию.

Вы, по всей видимости, недоумеваете, зачем я вам все это рассказываю?

А причина вот в чем: я хочу, чтобы вы помнили, покидая этот зал, что дистрибьютор Intel — это неотделимая часть нашей компании. Они члены нашей "семьи".

Наши дистрибьюторы в Советском Союзе не являются исключением. Они примут участие в наших образовательных программах и курсах переподготовки. Они будут осуществлять свою деятельность на том же качественном уровне, что и другие наши дистрибьюторы. Это означает, что наши клиенты в Советском Союзе будут получать столь же высококачественные продукцию и обслуживание, как и их коллеги в Европе, Соединенных Штатах Америки, Японии и других странах.

Дамы и господа, я чрезвычайно рад представить вам наших первых двух дистрибьюторов в Советском Союзе — СП "Диалог" и "Компьютерные технологии" (КоТ). Обе компании являются советско-американскими совместными предприятиями. СП "Диалог" было образовано в 1987 году и сейчас имеет более 80 филиалов с количеством сотрудников, превышающим 1000 человек. "КоТ" является очень молодой компанией, образованной в 1990 году. С американской стороны ее учредитель — компания "Мерисел", крупнейший в мире дистрибьютор компонентов для микроЭВМ.

Их обучение в качестве наших партнеров по бизнесу началось в январе этого года и будет продолжаться всегда.

Эти дистрибьюторы будут заниматься продажей изделий Intel на уровне 32-разрядных микрокомпьютерных платформ и систем высокой степени интеграции. Помимо плат и блоков они предложат другие изделия Intel, расширяющие возможности ПЭВМ, такие, как математические сопроцессоры Intel 386 (TM), модемы и системы факсимальной связи.

В настоящее время в Европе действуют 48 дистрибьюторов Intel. Сегодняшнее событие увеличивает их число до 50. Половина из них продает изделия Intel на уровне систем, другая же половина — компоненты.

Через несколько месяцев мы надеемся сообщить о нашем первом дистрибьюторе компонентов в Советском Союзе. Мы также намерены открыть этим летом свой офис в Москве. Здесь будет находиться наш постоянный представитель, в чьи функции будет входить оказание всесторонней поддержки нашим дистрибьюторам и заказчикам.

Присутствовал
на пресс-конференции
и брал интервью И. Липкин

Дэвид Смолл — человек, первым приступивший к изготовлению эмулятора Macintosh на Atari ST — представил на ганноверской ярмарке Cebit новую серию продуктов для улучшения пользовательских характеристик эмулятора Macintosh Spectre GCR.

Две крупнейшие новинки — дополнительная плата к Mega ST, позволяющая подсоединяться к локальной сети AppleTalk, и новая версия собственно эмулятора, поддерживающая неамериканские драйверы клавиатуры Mac, а также программы Dayna DOS Mounter и Apple File Exchange, полностью эмулирующие звуковые эффекты (что представляло проблему в прошлой версии) и поддерживающую мониторы с большим разрешением.

Кроме того, разработана плата MegaTalk, позволяющая использовать на Atari любые платы расширения, изготовленные для Macintosh, включая модемы, сетевые карты, факс платы и т.д.

Newsbytes News Network, March 15, 1991.

Фирма Toshiba выпустила портативный компьютер T3200SXC с цветным жидкокристаллическим экраном. Этот компьютер, весящий 7.7 кг, сочетает в себе мощность процессора 80386SX и активный матричный цветной дисплей размером 10 дюймов. Дисплей воспроизводит 256 цветов из палитры в 185000 оттенков. Так как дисплеи такого типа потребляют значительную мощность, этот "настольный портативный" компьютер питается от сети.

Newsweek, March 18, 1991.

Токио. В настоящее время фирма Hitachi Ltd. разрабатывает цветной плоский дисплей для рабочих станций. Матричный дисплей размером 10 дюймов (25.4 см) появится на рынке через 2 года и будет иметь разрешение 1120x780 точек. Дисплей будет в состоянии отображать одновременно 512 цветов из палитры 4096 цветов.

Время обновления экрана колеблется от 20 до 30 миллисекунд, а соотношение контрастности превышает 100:1. Дисплей имеет более 2.6 миллиона пикселов, каждый из которых несет в себе цветовую триаду.

Electronic Engineering Times, November 5, 1990.



МЕЖДУ ПРОЧИМ...

Немного о DiskTest

Неплохая программа проверки состояния диска DiskTest, входящая в пакет Norton Utilities Advanced Edition, позволяет не только обнаружить сбойные кластеры, но и перенести оказавшиеся на них файлы в безопасное место. Для этого от пользователя требуется одна дополнительная операция: на запрос программы о том, нужно ли отметить найденные сбойные кластеры как дефектные.

Однако проявление сбоя при чтении, в сущности, процесс случайный, поэтому любознательный пользователь наверняка заметит, что при разных проходах DiskTest находит различные сектора. Поэтому, чтобы повысить вероятность обнаружения опасных для работы мест на диске, лучше выполнить проверку неоднократно. В общем, достаточно элементарно написать командный файл, который будет в цикле вызывать программу DT.EXE, но все равно после каждого прохода, в котором удалось обнаружить сбойные области диска, пользователь должен будет подтверждать необходимость исключения их из дальнейшего использования.

Для решения этой проблемы следует использовать ключ /m при вызове DiskTest. В этом случае нечитаемые сектора без лишних вопросов будут отмечены как дефектные. Используя такую форму вызова DiskTest, можно написать командный файл, который будет проверять диск, пока вы не прервете его работу, нажав Ctrl-C. Ниже приведен его текст.

```
:label
DT/D/M
GOTO label
```

Запустите эту программу, уходя обедать, и это время не пройдет зря. Особенно, если вы давно не форматировали диск.

Компьютеры с процессором 80386SX почти не хуже настоящих 80386

Известно, что процессор 80386SX представляет собой упрощенный вариант “настоящего“ процессора 80386DX. Отличие заключается в использовании 16-разрядной шины при обращении к памяти. Внутренняя структура этих двух процессоров практически одинакова.

Поэтому отвечаем на вопрос о том, какой процессор выбрать для пакетов, ориентированных на 80386 (все чаще возникающий и у советских пользователей). Все режимы 80386SX аналогичны режимам 80386DX. Это относится и к работе с памятью. Отличие — в меньшем быстродействии компьютера на базе 80386SX (но и его цена значительно меньше!).

Поэтому любое программное обеспечение, написанное для 80386, будет успешно выполняться и на 80386SX. В том числе такие пакеты, как Windows 386, DESQview 386, трансляторы для 80386, и многие другие.

В то же время при выполнении данных пакетов на обычной IBM AT-совместимой машине, они не всегда работают как ожидается, возможны даже случаи зависания компьютера при попытках пакета использовать отсутствующие в процессоре 80286 ресурсы.

Еще один способ использования команды JOIN

Эта малоизвестная и нечасто используемая команда позволяет облегчить задачу резервного копирования дисков. Эту команду удобно применять с утилитами Fastback Plus фирмы Fifth Generation Systems или PC Backup фирмы Central Point Sotware, однако его не стоит использовать со стандартной командой BACKUP операционной системы MS-DOS.

Обычно такие пакеты требуют раздельного выполнения всех операций для каждого диска. Однако с помощью команды JOIN их можно выполнить для всех дисков сразу. Скажем, если ваш винчестер разбит на три раздела, для создания резервной копии всех дисков можно использовать такой командный файл:

```
JOIN D: C:\D_DRIVE
JOIN E: C:\E_DRIVE
{здесь нужно ввести имя вашей backup-программы с требуемыми параметрами}
JOIN D: /D > NUL
JOIN E: /D > NUL
```

Здесь команда JOIN описывает логические диски как отдельные каталоги на другом диске, что позволяет скопировать диск С:, заодно получив копии других дисков. Единственная хитрость заключается в том, что на диске С: действительно должны существовать описанные каталоги, причем они должны быть пустыми.

После того, как резервные копии будут готовы, две последние строки описанной выше программы восстановят исходное состояние системы.

Для восстановления резервных файлов подготовьте второй командный файл, в третьей строке которого нужно просто заменить параметры, чтобы обеспечить раскрытие копий.

Можно использовать аналогичную программу для глобального поиска файлов на всех дисках или для уничтожения, например, всех файлов с расширением .BAK или же для других групповых операций с файлами.

Ошибки вычислений в базах данных

Наш читатель В.Королевский из города Слоним Гродненской обл. обнаружил несколько особенностей СУБД dBASE III Plus, FoxBASE+ 2.00 и 2.01, REBUS. Надеемся, что эта информация заинтересует вас.

При использовании функций INT и MOD для обработки результатов каких-либо вычислений, бывают ситуации, когда эти функции возвращают неверный результат. Это случается при обработке чисел вида XXX.0000. Функция INT(n) может вернуть целое число на единицу меньшее ожидаемого, а функция MOD(n,1) — единицу.

Например, в результате вычисления INT(32.80-10.80) вы получите не 22, как можно было бы предположить, а 21. Столь же неожиданным для вас будет результат применения функции MOD к этому же выражению — он будет равен 1. Приведенная разность не уникальна. Аналогичный эффект вы можете наблюдать, обрабатывая, например, следущие выражения:

133.92-28.92
416.08-201.08

Кроме того, отметим, что некорректное поведение функций наблюдается на вполне определенных интервалах изменения операндов. Например, для числовых выражений:

32.80-10.80
33.80-10.80
. . . . . .
41.80-10.80
42.80-10.80

применение функций INT и MOD некорректно, а вот для:

30.80-10.80
31.80-10.80
43.80-10.80
44.80-10.80 и т.д.

вполне обычно.

Аналогично можно изменять значения второго операнда. Для разностей:

35.80- 3.80
35.80- 4.80
. . . . .
35.80-31.80

INT и MOD опять ведут себя безобразно, если же из 35.80 вычитать 0.80 или 1.80 или 2.80, как и 32.80, 33.80 и т.д., то все будет нормально.

Предлагаем вам подумать о причинах появления этого эффекта.

Если вы работаете с базами данных в одной из этих систем и пользуетесь этими функциями, то, чтобы ваша голова не болела, Королевский рекомендует применять их в следующем виде:

I = INT(ROUND(N,14))
I = MOD(ROUND(N,14),1)

Тестирование производилось на компьютерах IBM PC XT и AT с сопроцессором и без него, работающих с операционными системами DOS 3.30 и 4.0 и все было в порядке!

Новые ошибки от CompactSoft

Если вы считаете, что Norton Commander — система для начальников и студентов-второкурсников, если вы не можете смотреть на изображение в цветах "голубой на синем", то у вас нет (в этой стране) другого разумного выбора, кроме пакета XTree — по данным журналов PC World и PC Magazine — самого удобного и популярного в мире.

Недавно в пакете XTree Pro Gold версии 1.3 к удовольствию киевских хакеров нашлась первая и на сегодня единственная известная ошибка.

При просмотре содержимого файлов с помощью команды View в режиме шестнадцатиричного дампа вы никогда не увидите байт E6, на этом месте на экране будет написано E4, при этом в символьном поле справа текст отображается правильно. То же самое происходит при шестнадцатиричном редактировании файла. Более того, если вы в дамповом редакторе наберете

E6, на экране все же появится E4 и, опять-таки, верная буква в символьном поле.

Знатоки ассемблера могут сообразить, а для остальных сообщим, что E4 — команда IN, E6 — OUT.

Поэтому во избежание подобных случаев предлагается метод исправления этой ошибки. Сделать это можно тем же XTree Pro Gold (тот самый доктор, который может сам себя исцелить). В файл XTG_VIEW.XTP по смещению 6523h нужно вписать байт 36h. После такой экзекуции программа начинает работать лучше фирменной. Для полного счастья следует исправить еще и дамповый редактор XTG_HEXX.XTP. Ошибка в нем точно такая же. Найти заветное место, я надеюсь, сможет любой, исправивший первую.

И.Вязаничев
В.Королевский
Ю.Пронякин
Т.Шухова

По материалам:
J.Kamin "Combune Drives", PC/Computing, November, 1990.

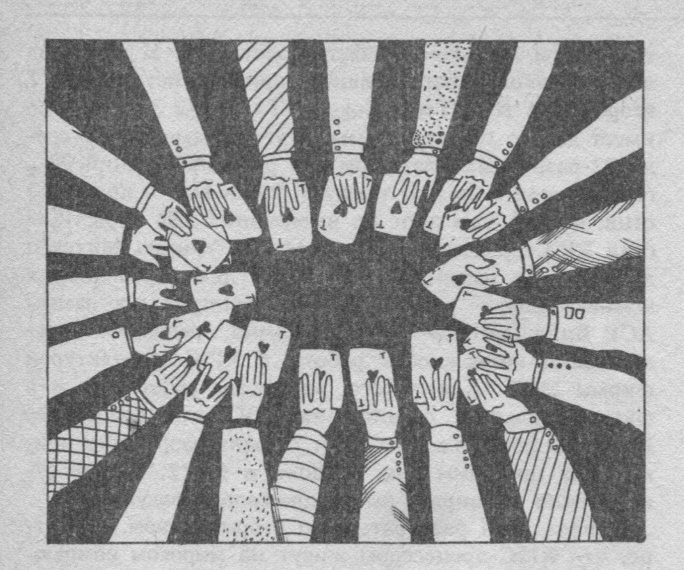
Сомнения, в каком направлении будут развиваться графические оболочки, овладевают умами пользователей компьютеров. Это сильно бьет по карману фирмы IBM: вместо OS/2 все покупают Windows.

Поэтому фирма провела ряд брифингов с пользователями, программистами, аналитиками и прессой о путях дальнейшего развития своей операционной системы.

Ключевым сообщением фирмы было то, что 32-битная версия OS/2 2.0 будет продаваться уже в конце этого года. Джеймс Каннавио, вице-президент IBM, назвал это будущее изделие "ДОС лучший чем ДОС, и Windows лучшие чем Windows" ("a better DOS than DOS and a better Windows than Windows.")

OS/2 в будущем сможет выполнять программы, написанные не только специально для нее, но и для DOS и Windows.

Newsbytes News Network, April 15, 1991.



"Опять этот UNIX", — проворчат недовольно любители MS-DOS. "Ну, наконец-то, UNIX", —скажут удовлетворенно приверженцы многопользовательских систем. Одно несомненно, — предлагаемая информация вызовет большой интерес у всех, кто неравнодушен к глобальным компьютерным проектам.

ДВАДЦАТЬ ИГРОКОВ РЕШАЮТ ПОЙТИ С ТУЗА

"Что это? Новый Крестовый поход против Советского Союза?", — спросил Чип Дейла.
"Вперед, Спасатели!", — изо всех сил закричала Гайка.

В знакомой всем компьютерным торговцам нашей страны формуле пользовательского счастья "покупайте Эй-Ти (AT) + И-ДжиЭй (EGA) + широкий принтер + винт на сорок" программные средства не удостоены даже упоминания. Ясно, что MS-DOS. А что же дальше?

Хотя дрейф в сторону графических пользовательских интерфейсов не обошел стороной и родные просторы, предчувствие надвигающегося путешествия к новым операционным системам еще не посетило душу простого советского человека. Более того, фирма Microsoft манит его перспективой скорого появления на рублевом рынке русифицированной и очень дружелюбно к нему, покупателю, настроенной версии системы MS-DOS 5.0.

Если же есть охота поиграться в воспоминания о будущем — к услугам эстетов и хакеров еще более дружелюбная (если у вас хватит денег на железо) операционная среда Windows 3.0.

И вот уже, услышав про требования к минимальной конфигурации для спокойной жизни с "окошками", покупатель издает сдавленный вопль и ошарашенно спрашивает: "А может, еще что-то найдется, подешевле малость?" До него доходит, что миграция в область операционных систем, в визитной карточке которых появляется титул "мульти...", дается очень недешево; гнаться за модой нелегко, как свидетельствуют о том коммерческие магазины. Так ради чего и в какой момент стоит пересчитать имеющуюся наличность и отправляться в плавание к новым берегам?

Многозадачность для человека, занимающегося индивидуальной трудовой деятельностью, — свойство туманное. Ветераны, разработавшие не одну систему р-р-реального времени на "Электронике-60", относятся к многозадачности a la Windows со скептицизмом и ядреным юмором деда Щукаря. Много всего понаворочено, а прерывание обработать — так замаессы.

Те же из ветеранов, шрамы которых своим появлением обязаны INMOS'у или DEMOS'у (в девичестве — ОС UNIX), не спешат переносить привязанности своей боевой молодости на настольную игрушку, будь она хоть с 486-м процессором. "Имидж не тот," — хмуро бормочут ветераны этого полка. "VAX — там куда ни шло, а на этой персоналке от UNIX'а проку никакого". Задиристая молодежь из малых и мельчайших предприятий попыталась было прельстить покупателя лейблом UNIX'а на "Эй-Ти +... (см. выше)", но пока безуспешно. Даже смелый бросок в 32-разрядное царство наперевес с застойной идеологией систем подготовки данных (ау, система "Клен") покупателя не

вдохновляет. "Это уж вы, ребята, перебрали, — говорят в один голос закаленные асушники, — Мы ведь еще СПД-9000 не продали, будь она неладна...".

Так что для советского анемичного рынка стремление бежать за паровозом по имени UNIX еще как-то не сформировалось. И тем не менее следить за маршрутом этого американского локомотива по запутанным колеям мирового компьютерного рынка приходится, а то момент упустишь. Вспомним, что рождался UNIX многопользовательской системой, и подивимся тому, как прочно укоренился он на рабочих станциях для индивидуалов. Вот уже и наш горделивый САПСАН с тевтонским характером несет в когтях лозунг "с UNIX'ом V.4 — хоть куда..."; но хозяевам этой птицы кажется, что подчинять UNIX капризам одного единственного пользователя расточительно, их рабочая станция — многопультовая. Этак и крыша поедет разбираться. Итак, что есть ОС UNIX — воплощение духа коллективизма или ближайшее будущее для владельцев персоналок с 486-м?

Двадцать фирм со всего света, собравшись под знаменем "Передовой компьютерной среды" (Advanced Computing Environment, ACE), объявили 9 апреля одновременно в трех компьютерных столицах — Нью-Йорке, Брюсселе и Токио — о создании открытой инструментальной среды для фирм-интеграторов и пользователей. Заявка не оригинальная, однако удивляет состав команды: здесь и тайваньский лидер (The Acer Group), и ветераны американского рынка (Control Data, Digital Equipment, Prime Computer), и солидные японские вкладчики (NEC, Sony), и осторожные европейские партнеры (Siemens Nixdorf, Groupe Bull, Olivetti Systems & Networks). Однако и эти зубры на поле не играют, отсиживаясь до времени на скамейке; в нападении — тройка фирм из молодняка: MIPS Computer Systems, Santa Cruz Operations (SCO) и Microsoft. Об остальных чуть позже, а сейчас расшифруем эмблему этого клуба. "Ace" по-английски значит "туз", тузов в колоде — четыре, как и положено.

Во-первых, козырной туз — открытость ко всему разумному, что накоплено компьютерными гуру из мира UNIX'а и MS-DOS. Минимальный стандарт POSIX (входной билет в мир совместимости с UNIX'ом) принят участниками этой команды безоговорочно. Участники программы ACE поддерживают скоординированное развитие как ОС UNIX (в ее реализации фирмой Santa Cruz Operations), так и системы OS/2 фирмы Microsoft. Пользовательская оболочка UNIX'а в принятом варианте известна покупателям SCO-UNIX'а под именем Open Desktop и ориентирована на рабочие станции, создаваемые для людей, привычных к программным продуктам Microsoft (да, это означает, что вы можете запустить прикладную задачу для MS-DOS на правах задачи в UNIX'е). А теперь... Слушайте нас внимательно, адепты Microsoft'а: у вашего новорожденного вундеркинда, системы OS/2, возникает раздвоение личности. Мало того, что у этого ребеночка сложились довольно странные взаимоотношения со своим коммерческим близнецом — Windows, так еще и юниксоидом быть захотелось. И приобрел отныне такой повзрослевший вариант системы OS/2 второе имя New Technology (NT). OS/2 3.0 (NT) не имеет дела с 16-разрядными платформами. А вот среди 32-разрядных платформ для рабочих станций 90-х годов нашел себе этот продукт эпохи еще и крестного отца, при родном папе по имени Intel X86. Крестным стал RISC-процессор фирмы MIPS. "Фирма Microsoft заботится о развитии стандартов для персональных компьютеров, — заявил на презентации программы ACE Билл Гейтс. — Эта программа предлагает пользователям возможность работы с RISC-архитектурой фирмы MIPS. Кроме того, мы будем заботиться и о функциональных расширениях ОС для владельцев Intel-платформ. Те, кто ведет сегодня разработку прикладных программ для Windows и OS/2, смогут существенно расширить рынок для своих продуктов."

Теперь пора разобраться со вторым тузом. Известно, что RISC-процессоры живут на мировом компьютерном рынке трудной, но интересной жизнью. Для начала отметим, что у массовых пользователей RISC-компьютеров пока немного (полторы — две сотни тысяч). Эти платформы в большинстве своем стали инженерными рабочими станциями, которые не менее чем вдвое дороже "хороших" персоналок. Появился RISC-процессор и у фирмы Intel (i80860), но пока он на самостоятельность в платформах рабочих станций претендует редко, оставаясь на ролях сопроцессора. Разборка на RISC-рынке идет в основном между двумя архитектурами — SPARC и MIPS. Для рабочих станций ОС UNIX стала наиболее ходовой за последние три-пять лет, и такой альянс подкреплен появлением группы стандартов (X.11, SVID, NFS, ANSI C и уже упомянутый POSIX). Так что с открытостью — первым тузом в инициативе ACE — дела обстоят хорошо. Мобильность UNIX'а доказана многажды, и все же мобильность — это не совместимость. Поэтому поставщики RISC-платформ с UNIX'ом предлагают более или менее дешевые лицензии на спецификации своей аппаратуры. Сложившийся на базе такой лицензии отряд фирм Sparc International обеспечил себя несколькими поставщиками процессоров SPARC, но эти процессоры не совместимы по контактам своих микросхем. Среди ведущих фирм на призыв вступить в группу Sparc International откликнулись пока 4. Во вновь созданной команде ACE таких солидных партнеров собралось десятка полтора... Фирма MIPS предлагает сегодня лицензию на спецификацию ARC (Advanced RISC Computing), охватывающую процессорное ядро, магистраль, набор малых интерфейсов, формат и цветность графического растра, а также расположение клавиш ("де-факто" стандарт клона IBM PC со 101 клавишей) и форматы файлов. Такой всеобъемлющий стандарт на RISC-платформу, да еще с участием фирм первой десятки компьютерного мира, имеет серьезные шансы на успех.

Третий туз в колоде — сохранение облика UNIX'а как многопользовательской системы. Дебют SCO-UNIX под именем Xenix состоялся с подачи фир-

мы Microsoft на платформах с процессором i80286, для которых пользователи хотели иметь возможность работы с небольшим числом ASCII-терминалов. В сбыте таких платформ преуспела, например, фирма Altos. Развитие UNIX'а для Intel-платформ фирма Santa Cruz Operations прочно захватила в свои руки, и сегодня число продаж SCO-UNIX составляет сотни тысяч. Участники программы ACE приняли решение развивать OS/2 как однопользовательскую систему, предусмотрев такие направления развития, как поддержка многопроцессорных архитектур, многоуровневой защиты общих ресурсов и встроенной коммуникативности (способности взаимодействовать с различными сетевыми архитектурами). Другим решением за системой SCO-UNIX закреплена область многопользовательских систем с поддержкой симметричных многопроцессорных архитектур, коммуникативности (в том числе с сетевыми архитектурами DECNet и SNA) и модели "клиент-сервер" при работе в интегрированных локальных и региональных сетях. В этой реализации UNIX'а обязательным становится соблюдение таких стандартов, как OSF AES, X/Open XPG3 и SVID Issue 2 (и, разумеется, общего с OS/2 стандарта POSIX 1003.1). Участие в команде фирмы DEC делает правдоподобным утверждение о координации развития систем SCO-UNIX и ULTRIX. О реализации графической оболочки в составе SCO Open Desktop на равных правах с Windows и Presentation Manager уже упоминалось, так что и здесь следует ожидать быстрого сближения.

Такое перекрестное опыление четырех саженцев в компьютерном саду — систем OS/2 и UNIX, платформ MIPS RISC и Intel x86 — должно привлечь разработчиков прикладных программ и модулей расширения для рабочих станций, серверов и классических многопользовательских систем с терминалами. Есть надежда, что гибридные плоды не заставят себя ждать; позиции Microsoft и Santa Cruz Operations на рынке программных средств сильны (35000 и более 4000 программных продуктов в фонде для каждой линии соответственно), а такие фирмы, как DEC, Silicon Graphics, Siemens Nixdorf, Olivetti и Groupe Bull, способны внести в фонд прикладных программ свои продукты с устоявшейся репутацией на отраслевых и региональных рынках. Симптоматично участие в программе ACE фирм, имеющих широкую известность в узких кругах. Например, фирма Tandem Computers лидирует на рынке отказоустойчивых систем для банков, транспорта и связи; фирмы Prime Computer и Control Data занимают устойчивые позиции на рынке САПР; фирма Wang Laboratories лидирует на рынке систем для научных исследований. Поэтому четвертым тузом в колоде программы ACE безусловно является присутствие в этой команде фирм, знающих, что такое "вертикальные сегменты рынка", на отечественном новоязе получившие название проблемно-ориентированных систем. Добавим, что с объявлением программы ACE существенное подкрепление на компьютерном рынке получили магистрали EISA и TURBOChannel, что для них совсем не лишнее в условиях конкуренции с магистралью Microchannel фирмы IBM. Ставка, покрываемая этим четвертым тузом — 50 миллиардов долларов; именно этим числом оценивается совокупный годовой оборот фирм — участниц ACE.

Итак, исходные позиции программы ACE:

- наличие нескольких поставщиков для RISC-процессора MIPS, выпускающих совместимые по контактам СБИС;
- переход к массовым поставкам как SCO-UNIX, так и OS/2 NT в виде продуктов полиграфического класса (под этим неуклюжим определением авторы понимают жаргонное американское "shrinkwrap software" — "программный продукт в термоусадочной пластиковой оболочке" — т.е. программы, распространяемые в розничной торговой сети);
- участие ведущих фирм компьютерного мира;
- опора на фонд прикладных программ для UNIX и Windows;
- работа со всеми каналами сбыта, включая сбыт фирмам — интеграторам прикладных систем, независимым фирмам — поставщикам программ и аппаратуры, а также массовым пользователям через розничную сеть.

Еще одна цитата из выступления на презентации программы ACE в апреле 1991 года. "Мы считаем, что программа ACE даст дорогу лавине технических инноваций и конкретных выгод для пользователей по примеру эпохи становления индустрии персональных компьютеров... Положено начало быстрому распространению систем нового поколения в стандарте ACE для наиболее важных сегментов компьютерного рынка в 90-х годах." Автор этих слов — партнер ACE, президент фирмы Compaq Computer Род Кэнион. Он вместе с Биллом Гейтсом, Кеннетом Олсеном (президент фирмы DEC), Дагом Майкелсом (президент SCO) и Робертом Миллером (президент MIPS) выступил поручителем этой амбициозной программы.

Для любителей технических подробностей стоит сообщить некоторые характеристики "рабочей лошадки" программы ACE — RISC-процессора R4000. Этот процессор имеет 64-разрядную ширину тракта обработки данных и включает в себя центральный процессор, процессор целочисленной арифметики, процессор арифметики с плавающей точкой, диспетчер памяти и кэш-память нижнего уровня; реализация всех перечисленных составляющих проведена в одном корпусе СБИС. Фирма MIPS не занимается производством СБИС, предоставляя лицензию внешним поставщикам; сегодня в их число входят NEC, Siemens Nixdorf, LSI Logic Integrated Device Technology и Performance Semiconductors.

Важной особенностью архитектуры MIPS RISC-процессоров, позволяющей радикально повышать производительность, является "суперконвейеризация вычислений" ("superpipelining") — схема спаривания команд, подаваемых в конвейер команд на каждом такте работы процессора. 64-разрядная магистраль данных мо-

жет показаться экзотикой, однако разработчики фирмы MIPS утверждают, что это техническое решение абсолютно неизбежно для архитектур 90-х годов. Задачи компьютерной графики вплотную подошли к пределу 32-разрядной адресной сетки. Ширина внутренней магистрали кэш-памяти в R4000 и вовсе составляет 128 разрядов, что необходимо для согласования всех схем управления конвейером.

Для читателей, настроенных на решительные деловые предложения, сообщаем, что 11 апреля в Москве подготовлено для прессы следующее совместное заявление:

"Фирма SCO и совместное предприятие Future Technologies International (FTI) объявляют, что они достигли договоренности, согласно которой FTI выступает в роли ведущего дистрибьютора программных продуктов Santa Cruz Operations в СССР. До недавнего времени программные продукты SCO, получившие распространение в СССР, поставлялись легально западными фирмами только в составе систем "под ключ". Получили также хождение версии некоторых продуктов SCO из различных "неформальных" источников. Путем установления контактов, подобных объявленому, фирма SCO рассчитывает придать регулярный характер каналам сбыта своей продукции". Речь идет о поисках дилеров по отраслям и регионам, заинтересованных на первых порах в оснащении платформ класса i386/486 системой SCO UNIX/Open Desktop и различными инструментальными программными средствами для АСУ, САПР и САНИ. СП FTI, партнером которого является австрийская фирма Future Technologies (торговый дом, специализирующийся в области высоких технологий), заинтересовано также в сотрудничестве по локализации программных продуктов SCO.

А. Гиглавый, К. Чащин



НОВОСТИ

Японская фирма Kyocera объявила о своих намерениях начать продажи в США и Европе своего миниатюрного компьютера Refalo с рукописным вводом текста.

Refalo — это комбинация компьютера размером A5 и записной книжки. Весящий всего 650 грамм, он выполнен на базе процессора NEC V30, имеет прошитую в ПЗУ MS-DOS 3.22, а также последовательный порт для связи с другими машинами.

Информация в компьютере хранится на специальных магнитных карточках. В комплекте с компьютером продаются записанные на таких носителях текстовый редактор, дневник, записная книжка, часы и изощренный калькулятор. Фирма сейчас разрабатывает еще одну карточку, которая позволит осуществлять обмен небольшими сообщениями по радио.

Фирма заявляет, что начинает разработку делового программного обеспечения, ориентированного на европейский рынок. Готовый продукт вместе с программами появится в продаже осенью этого года в Европе и в начале следующего года — в США.

Newsbytes News Network, April 17, 1991.

В планы компании Apple входит образование отдельного подразделения для работы на рынке СССР и Восточной Европы. Штаб-квартира Apple Central Europe будет находиться в Варшаве.

Фирма планирует начать продажу своих компьютеров на рынках бывших социалистических стран, как только закончит весьма непростой процесс локализации программного обеспечения. По мнению многочисленных представителей фирмы, выходить на рынки без готового софтвера — только себе вредить.

Парижский представитель фирмы сообщил, что окончание соответствующих работ в разных странах планируется примерно на одно и то же время — лето этого года. Сразу за этим последуют объявления о начале продаж.

На западе компания сильна в области поставок машин школам и университетам. В СССР ей придется побороться за этот рынок с IBM и рядом других компаний.

Кстати, представители Apple гордо заявляют, что на столе у Президента Горбачева стоит Macintosh.

По оценке Московского представителя Apple Грега Борровски, сейчас в СССР имеется не более тысячи компьютеров Macintosh. Тогда как, по утверждению Хейки Аувинена, финского консультанта по советской электронной промышленности, IBM-совместимых машин в стране насчитывается от одного до полутора миллионов, причем ближе к одному, чем к полутора.

Newsbytes News Network, April 15, 1991.

Фирма Ashton-Tate начала поставку программы dBASE IV RunTime Plus, которая позволит работать с любыми прикладными системами, написанными на dBASE IV, версия 1.1, на компьютере Macintosh. Цена пакета 195 долл.

Теперь dBASE существует в четырех различных средах — под MS-DOS, на компьютерах Digital Equipment VAX с ОС VMS; на рабочих станциях Sun Microsystems с версией системы Unix — SunOS; и на компьютерах Macintosh.

Кроме предоставления простого средства выполнять программы, RunTime Plus имеет дополнительные возможности для создания, модификации и отладки dBASE программ на компьютере Macintosh.

Newsbytes News Network, April 10, 1991.

Цветной термопринтер выпущен канадской фирмой CalComp. Он печатает на бумаге стандартного формата А3 с разрешающей способностью 300 точек на дюйм.

Краска при нагреве переносится со специальной 4-цветной (зеленый, желтый, красный, черный) термочувствительной ленты на бумагу по одному цвету за проход печатающей головки. Также поставляются черные одноцветные ленты для быстрой однопроходной печати. По заявлению фирмы, принтер может воспроизводить "тысячи цветов".

Полностью совместимый с Adobe PostScript, принтер может подключаться к компьютеру через интерфейсы AppleTalk, RS-232, SCSI или параллельный порт. В одну из моделей встроен алгоритм преобразования векторной графики в растровую, что упрощает его использование в системах САПР.

Newsbytes News Network, April 11, 1991.

Девятнадцатилетний нью-йоркский хакер Марк Абене, известный под кличкой Phiber Optik, был приговорен судом к 35 часам коммунальных работ за доступ к компьютеру без разрешения.

Абене был арестован 5 февраля полицией штата Нью-Йорк и обвинен во вмешательстве в работу компьютера и проникновение в него без разрешения, что по законодательству США считается уголовным преступлением. Обвинения были сняты, когда обвиняемый чистосердечно признал себя виновным в "незаконном использовании компьютера". Приговор положил конец году неопределенности в его судьбе — ордер на изъятие его компьютера был выдан в январе 1990 года.

Абене заявил, что отработает требуемое наказание в местном госпитале.

Newsbytes News Network, April 8, 1991.



МАЛОЕ ПРЕДПРИЯТИЕ "ИНФОРМАТИКА"

УЧРЕДИТЕЛЬ - Институт проблем информатики Академии наук СССР

Логическое программирование

Объектно-ориентированное программирование

Инструментальные средства

ПРОГРАММНЫЕ ПРОДУКТЫ. ЛЕКЦИИ, КОНСУЛЬТАЦИИ. СОЗДАНИЕ ПРИКЛАДНЫХ ПРОГРАММ

Текстовые процессоры

Операционные системы

Стохастические системы

117900 Москва, В-334, ул. Вавилова, д. 30/6, ИПИ АН СССР, МП "Информатика"
Телефон: (095)-362-46-54, факс: (095)-310-7050, телекс: 411853 INFO SU

Цена 3.15